Air Force Institute of Technology
AFIT Scholar

Theses and Dissertations | Student Graduate Works

9-2023

Analysis of Multi-agent Routing Solution Methodologies Exploring a Mosaic Warfare Strategy

Stephen D. Donnel

Follow this and additional works at: https://scholar.afit.edu/etd

 Part of the Operations Research, Systems Engineering and Industrial Engineering Commons

Recommended Citation

Donnel, Stephen D., "Analysis of Multi-agent Routing Solution Methodologies Exploring a Mosaic Warfare Strategy" (2023). *Theses and Dissertations*. 7664.
https://scholar.afit.edu/etd/7664

This Dissertation is brought to you for free and open access by the Student Graduate Works at AFIT Scholar. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of AFIT Scholar. For more information, please contact AFIT.ENWL.Repository@us.af.mil.

ANALYSIS OF MULTI-AGENT ROUTING SOLUTION METHODOLOGIES EXPLORING A MOSAIC WARFARE STRATEGY

DISSERTATION

Stephen D. Donnel, Capt, USAF

AFIT-ENS-DS-23-S-014

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

Distribution Statement A
Approved for Public Release; Distribution Unlimited.

The views expressed in this document are those of the author and do not reflect the official policy or position of the United States Air Force, the United States Department of Defense or the United States Government. This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

AFIT-ENS-DS-23-S-014

ANALYSIS OF MULTI-AGENT ROUTING SOLUTION METHODOLOGIES EXPLORING A MOSAIC WARFARE STRATEGY

DISSERTATION

Presented to the Faculty

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

Stephen D. Donnel, BS, MS

Capt, USAF

September 14, 2023

Distribution Statement A
Approved for Public Release; Distribution Unlimited.

AFIT-ENS-DS-23-S-014

ANALYSIS OF MULTI-AGENT ROUTING SOLUTION METHODOLOGIES EXPLORING A MOSAIC WARFARE STRATEGY

DISSERTATION

Stephen D. Donnel, BS, MS
Capt, USAF

Committee Membership:

Dr. Brian J. Lunday
Chair

Capt Nicholas T. Boardman, PhD
Member

Maj Brigham A. Moore, PhD
Member

Abstract

Recognizing that communication between assets may be possible locally but not globally (e.g., due to disruptions to a communication network), Mosaic Warfare requires the movement and operation of multiple, dispersed assets in smaller groups (i.e., *tiles*), within which exist hierarchical, functional relationships between assets. This research sets forth and evaluates a Hierarchical Asset Tiling and Routing Heuristic (HATRH) to implement Mosaic Warfare for an enterprise of aerial assets comprised of airborne sensors, command and control aircraft, and strike aircraft seeking to move towards and destroy a set of stationary targets. The HATRH is comprised of three, iteratively applied algorithms: a grouping algorithm to group assets into functional tiles, and two algorithms respectively related to group movement and individual asset movement. Embedded within the latter two algorithms are user-determined parameters that roughly correspond to group and individual asset agency within the mosaic. Extensive testing examined the effect of these parameters and asset density for three different operational scenario designs, and with comparison to optimal (i.e., efficient) asset utilization via two Price of Anarchy (POA) inspired metrics. Results showed the user-defined parameter corresponding to individual asset agency notably influenced both average munition expenditures and the average distance traveled by assets. In the scenario wherein assets initially surround adversary targets, both the individual and group agency user-defined parameters influence operational efficiency, in terms of munitions expended and fuel consumed.

Proceeding, the next research examines the problem of routing multiple assets of different types over a network to service demands in a collaborative manner. The servicing is collaborative in that, when servicing a demand, the different types of

assets must do so nearly simultaneously. Moreover, whereas some asset types must service demands by visiting them, other asset types may provide service proximally. This study sets forth a mixed-integer linear program to model this variant of a vehicle routing problem. In addition to directly solving problem instances via a commercial solver, this research proposes two permutations of a model decomposition heuristic, as well as two preprocessing techniques to impose instance-specific bounds on selected decision variables. Comparative testing evaluates nine combinations of solution methods and preprocessing options to solve a set of 216 instances that vary significant parameters. Results manifest trade-offs between the likelihood of finding a feasible solution with bounded computational effort and the relative quality of solutions identified. For larger networks, the preprocessing technique leveraging a nearest neighbor heuristic in combination with any solution method most frequently identified feasible solutions for the set of test instances (i.e., $\sim$90% of instances), with lesser solution quality (i.e., within 15% of the best solutions identified, on average). Worst performing for larger networks was a model decomposition technique that first routes assets providing service proximally, and omitting either preprocessing technique; although this combination yielded the best solutions when it identified a feasible solution, it only did so for $\sim$55% of instances. Other solution method performances exhibit noteworthy nuance, as detailed herein.

Finally, research examines the problem of routing multiple assets of different types over a network to service demands, where the demands must be serviced by asset types in sequential order within a bounded amount of time, and minimizing the cumulative service time is of interest. More specifically, this research seeks to identify effective network disruption strategies with limited resources to maximize the minimal cumulative service time. Within a bilevel programming structure for this Stackelberg game, the upper-level problem determines the disruption strategy, and the lower-level prob-

lem routes the assets. This research considers and tests three soluton procedures: a greedy construction heuristic (GCH) that iteratively identifies each disruptive action, a customized implementation of simulated annealing (SA), and an enhanced variant thereof (eSA) that leverages a prioritized identification of candidate solutions along with a tabu list. Testing compares the solution methods on similar instances over a range of selected algorithmic and instance-specific parameters. Results showed the enhanced simulated annealing method to perform best, and extended testing explored the effect of increasing selected problem sets on the relative improvement of eSA over GCH, as well as its effect on algorithmic runtimes.

Dedicated to my wife whose love and support are more infinite than the real numbers. To my daughter and sons - may your quest for knowledge have no limit.

Acknowledgements

I want to express my utmost appreciation to my advisor Dr. Brian Lunday, for his guidance throughout the development of this research. Another expression of gratitude to committee members Dr. Boardman and Dr. Moore for their assistance and input allowed this research to mold into its intended vision and application. Their combined direction and mentoring made this body of work feasible. Finally, a sincere thank you to my wife, who kept me focused, provided unparalleled encouragement, and made everything possible.

Stephen D. Donnel

Table of Contents

Page

List of Figures

List of Tables

ANALYSIS OF MULTI-AGENT ROUTING SOLUTION METHODOLOGIES EXPLORING A MOSAIC WARFARE STRATEGY

I. Introduction

1.1 Motivation and Background

Defense analysts for decades have theorized that enemy forces would employ strategies in anti-access to hinder the deployment of U.S. forces and assets and to disrupt communication amongst command structures. China's military investments affirm their status as a potential adversary with the ability to utilize such techniques that limit the locations of U.S. forces or compel them to operate at a greater distance from the focal point (e.g., targets) within a kinetic conflict (Cliff et al., 2007). Adversaries of the U.S. can asymmetrically focus weapons and concept developments on attacking vulnerable nodes in U.S. military operations (Dougherty, 2019). For instance, China intends to utilize anti-access/area denial (A2/AD) measures to achieve strategic effects that disrupt U.S. operations by making critical communications, command, and control networks ineffective (Deptula et al., 2019). Degrading the U.S. military's ability to operate and communicate efficiently portends an unraveling of security guarantees by the United States to its allies and partners. In turn, undermined confidence in U.S. security assurances can undermine these relations critical to the established global order (Dougherty, 2019).

Deptula et al. (2019) stresses the importance of military planners differentiating between the warfighting strategies of Russia and China, despite their common A2/AD design methodology focusing on countering America's military power. Russia plans

to use A2/AD systems in a war of attrition, protecting its operations from external threats or intruders (Barrie, 2019). Russia has demonstrated during military exercises the integration of long-range surface-to-air missiles (SAMs) and medium-range ballistic missiles, effectively creating an aerial minefield for enemy aircraft attempting to penetrate Russia-controlled airspace (Cabot, 2018). The past decades have shown China increasing capabilities of its core A2/AD technology, including ground and air defense systems, next-generation tactical aircraft, electronic warfare, and high-precision missiles (Yevtodyeva, 2022). China has adopted a strategy poised at using A2/AD to attack the network and communications elements of the U.S. military forces. By preventing command-and-control (C2) and Intelligence, Surveillance, and Reconnaissance (ISR) assets from getting close to a region of active conflict, China can significantly degrade the capability of the U.S. system (Deptula et al., 2019; Fravel, 2016). A conflict with China would put U.S. forces against the Chinese "system-of-systems paralysis" strategy, designed to target an adversary's C2 and interrupt the American battle network at all levels (Costello et al., 2016; Deptula et al., 2019; Pickrell, 2019).

Where sharing information via a communication network has been extolled as a virtue, adversaries have evolved to exploit this reliance by attacking key communication nodes in the U.S. force network, negating their ability to maneuver and communicate effectively (Colby, 2019; Deptula et al., 2019). Moreover, these global powers have observed the U.S. military's strategy as being both reluctant and slow to change from the "Desert Storm model," where any opponent could be dominated by leveraging the massive technical advantages (e.g., integrated communication network, global positioning system, precision-guided munitions) the U.S. began developing in the 1970s. This strategy held abundant before the 2010s when the U.S. lacked a contemporary military competitor. Now, A2/AD systems can create exploitable openings

in the current way the U.S. fights, creating vulnerabilities to attack with rapidly deployable and heavily armed conventional forces (Colby, 2019). The advancements in technology and strategy by China and Russia now threaten critical infrastructure in U.S. and allied bases, air defense, and critical logistical nodes. Modern force structures rely on sharing information in real-time to update forces with better operational awareness to close kills chains (Deptula et al., 2019).

The changing environment of war has forced the United States to adopt new evolutionary strategies and tactics to prevail in future peer-to-peer contests. The most significant threats to the security interest of America and its allies are the growing powers of China and Russia and their revisionist ambitions. Chinese and Russian strategies use A2/AD against the United States and their allies to prevent any intervening attacks against their aggressions (Deptula et al., 2019). Future wars will focus on information and decision-making capabilities. The battle network will concentrate on electronically and physically degrading an opponent's ability to obtain accurate information while introducing false information, destroying their ability to orient and attack or defend. Technology advancements in artificial intelligence (A.I.) also force the need for more advanced strategies and warfare. Decision-centric warfare may become the most effective method of exploiting A.I. and autonomous systems, quickly emerging throughout the battlefield. Defense Advanced Research Project's Agency's (DARPA's) decision-centric warfare example, Mosaic warfare, centralizes itself on many components of manned and autonomous units, guided by human control, can adapt to complex situations and dispute enemy centers of gravity, preventing further aggression (Clark et al., 2021). Sensing and targeting grids will need to be able to find, identify, and track any vehicles associated with enemy invasion or threat of communications jamming. Additionally, grid changes must adapt autonomously, nominating and guiding systems where links are temporarily severed (Ochmanek,

2022).

Mosaic Warfare owes its namesake to how smaller force structures elements can be rearranged into different configurations, similar to how mosaic artwork comprises different shaped tiles that form an overall image. Conceptually, a Mosaic Warfare force employs disjoint assets and platforms to formulate an operational system without requiring a single C4I system that networks all components. The artwork, or operational warfare design, is the larger picture resulting from the decentralized orchestration – an oxymoron, to be sure – of the capabilities of platforms like combat aircraft, including radar, fire control, and missiles (Deptula et al., 2019). The individual pieces of the mosaic come together to form the complete picture, in this case, a force plan. Burns, the previous director of DARPA's Strategic Technology Office, merits credit for advancing the concept of Mosaic Warfare within the U.S. Department of Defense. He identifies a key strategy of the Mosaic Warfare design: to overwhelm enemy forces by utilizing large amounts of sensor and weapon platforms. Burns said, "When you attack in parallel across a wide front and you have distributed your sense-decide-and-act systems across a wide number of platforms, you can mass your firepower without having to mass your forces" (*DARPA Tiles Together a Vision of Mosaic Warfare*, 2020).

Currently, analysts are researching systems warfare from the view of the Chinese People's Liberation Army's (PLA) understanding of systems on systems warfare, including the current warfare tactics and planner military strategy (Engstrom, 2018). Outlined are four types of targets that PLA planners will seek to strike through kinetic or non-kinetic attacks to incapacitate a foe's operation systems: (1) degrading the flow of information, (2) target key nodes or functionalities, (3) disrupting the operational architecture of the enemy's operation system, and (4) disruption the time sequence of adversary's tempo (Engstrom, 2018). Given its potentially disruptive

nature against the U.S. military's hegemony, systems confrontation may evolve into the new warfighting tactic adopted by many and not limited to China. In the absence of change to their current employment concepts, U.S. forces cannot successfully prosecute the goals outlined in the *National Defense Strategy* (Mattis, 2018). The innovative employment concepts of Mosaic Warfare utilizes existing U.S. forces and better employ them for the system warfare of the future (Deptula et al., 2019).

Mosaic Warfare proposes more diverse courses of action (COA) to a force, combined with command and control (C2), that provides options for rapid, decentralized decision-making to make fast and effective decisions. The effectiveness of Mosaic Warfare arises from the force's disaggregated structure, combined with the use of human command and machine control, complicating an opponent's decision-making process into an insoluble collection of dilemmas (Clark et al., 2021). The concepts behind Mosaic Warfare seek to eventually address the current force's shortcomings. These shortcomings include but are not limited to small inventories of capable high-end multi-function platforms, long development time to field major weapon systems, and current force design not being able to withstand attrition (Deptula et al., 2019). In addition, enemy forces are becoming more aggressive, with China expressing the intent to be able to take Taiwan by force by the year 2027 (LaGrone, 2021). The Mosaic Warfare force design aims to gain offensive initiative against the enemy while remaining adaptive throughout the military operational spectrum to create a kill web of numerous components. These components become designed to minimize targetable friendly nodes while ensuring the military unit remains effective in a contested environment (Deptula et al., 2019).

The rebuttal for a systems warfare strategy is to remove single points of failure, avoid the single data link, and strategize away from a standard operational formation in which enemy forces can concentrate their plans. A force employing Mosaic War-

fare creates this counter, networking together disaggregated forces that to increase effectiveness of U.S. forces, expand the kill paths, which are the paths necessary for allied assets to reach desired targets, and leave the adversary unable to establish its next target decisions (Deptula et al., 2019). Burns summarized Mosaic Warfare well, characterizing the dynamics of interconnected systems by stating, "Why don't we take simpler systems and then network them together, have them share, collaborate sense their world in their own unique way – and put them together?" (*DARPA Tiles Together a Vision of Mosaic Warfare*, 2020).

Mosaic Warfare creates a comprehensive model for systems whose attributes can help increase the speed of action across the U.S. military enterprise. Moreover, the principles and technologies that develop a force design deliberately tailored for Mosaic Warfare will enable the U.S. to stay competitive against adversaries in future system warfare. The fundamentals of Mosaic Warfare entail the disaggregated information sharing related to radar, fire control, and missiles into smaller elements rather than hosting them on a common platform for centralized command and control. This transforms the kill chain from a linear kill chain to a networked web of individual components making up the larger operations design – the mosaic (Deptula et al., 2019).

However, current weapon systems are not designed to operate this way, preventing a transition to a Mosaic Warfare strategy. The weapons of the modern military are designed to fit into a puzzle, similar to a jigsaw piece, designed to fit into one place of need, not the tile of a mosaic, designed to be interchanged and reassembled as the need of the battlefield alters and changes (*DARPA Tiles Together a Vision of Mosaic Warfare*, 2020). The future of the U.S. force design needs to be adjusted to overcome future challenges and current shortfalls. First, the U.S. military has become reliant on a disproportionate number of competent, multi-functioning platforms serving as

valuable adversary targets. Losing a small number of these aircraft is detrimental to the current operational structure (Deptula et al., 2019). Second, the current buying trend of the U.S. military has become vastly inefficient, where the fighter aircraft fleet has halved, and the bombers dropped to a fifth of what they were two decades ago (Kass, 2019). Third, as specialized and robustly engineered machines are designed, the U.S. military needs to plan on buying them in more significant numbers to meet the demand of a future system's war eventually. Finally, the developmental process and time frame to create and field new technologies necessary for tomorrow's battlefield is too long. The problem is beyond an engineering or technological problem, but one of bureaucratic processes and acquisition schedules (Deptula et al., 2019; Schwartz, 2010).

1.2 Research Objective and Scope

Mosaic Warfare will require more than new technologies. It will also require a study of its operational concepts to examine the effectiveness of smaller, adaptive command and control structures to direct subsets of assets within missions. Additionally, it will require a complex understanding of routing and movement dynamics and the protocols necessary for a semi-autonomous fleet of vehicles with specialized capabilities to collaborate and move throughout a network or spatial region. Overall specific objectives of this research are the ability to detect and observe targets with Intelligence, Surveillance, and Reconnaissance (ISR) aircraft, process intelligence, and direct action via command and control (C2) aircraft, and attack targets using strike aircraft. Some of the aircraft may be multi-role, but all of the aircraft will fly in a network-contested environment, so it is of interest to both centrally plan their routes and adapt those plans in a decentralized manner when anticipated conditions change and disruption within the C2 network, leaving communication only possible between

relatively proximal assets.

This research will explore routing multiple vehicle types with specialized roles vis-á-vis a Mosaic Warfare style decentralized operation with centralized control. The first objective is to investigate the dynamic between assets moving as a group, or tile in a grand mosaic, and their individual asset agency, exploring how movement limitations affect other factors in the operations such as munitions expense and fuel cost. Next, determine solution methods to develop and explore solution discovery and quality of routing multiple agents in a collaborative environment as commercial solvers struggle to find feasible solutions in larger operational instances. A subsequent objective investigates the relationship between assets attempting to satisfy demand on predetermined target locations while traversing a contested network, where an adversary can delay the routing and movement speeds of assets.

1.3 Organization of the Dissertation

Within the remainder of this paper, Chapter II presents a Hierarchical Asset Tiling and Routing Heuristic to implement Mosaic Warfare for an enterprise of aerial assets, which has two user-determined parameters embedded within to roughly correspond to group and individual asset agency with the mosaic. Chapter III develops a mixed-integer linear program to route multiple asset type vehicles collaboratively, in that they much service demand in a near simultaneous manner, wherein some asset types require direct delivery to defined demand node and other asset types provide service proximally. Next, Chapter IV examines a bilevel problem to model a Stackelberg game, wherein the lower-level problem minimizes cumulative servicing times while routing assets. At the same time, the upper-level simultaneously adapts a strategy to impose a limited number of disruptive actions to slow down multiple assets being routed on the network. Finally, Chapter V concludes the research and suggests

meaningful future excursions.

II. Analysis of a Distributed Command and Control Algorithm to Implement Mosaic Warfare

2.1 Introduction

Within the global community, the United States (U.S.) has the greatest expenditure on its armed forces (Szmigiera, 2022) and is habituated to the idea that sheer military superiority will deter conflict. However, recent conflicts and enemy strategic performances have shown that a future war involving the U.S. may be inevitable, and a military loss could be possible (Dougherty, 2019). To win, the U.S. must focus on revitalizing its warfighting strategy because potential adversaries are carefully designing warfare systems to counter its contemporary methodology of warfare that relies on secure, networked communications (Fravel, 2016).

The U.S. military has become increasingly reliant on the network warfare it has been building in the past years, confident that information sharing would remain present throughout the network, enabling the centralization of operational control. Moreover, the lack of a peer adversary for many years established the belief that U.S. forces would be able to retain control over all operational domains, including communication's cyberspace. Such confidence developed the foundation for an information network philosophy of conflict, one that inadvertently adopted an enormous amount of vulnerability (Deptula et al., 2019).

The most significant threats to the security interest of America and its allies are anti-access and area denial (A2/AD) strategies that prevent forces from entering into or conducting sustained actions within an area of operations (Deptula et al., 2019). Such A2/AD strategies threaten communications networks (Mulvenon et al., 2006), and future wars will focus on information and decision-making capabilities. The changing environment of war has forced the U.S. to consider new strategies and

tactics to prevail in future major theater conflicts.

Network warfare will concentrate on electronically and physically degrading an opponent's ability to obtain accurate information while introducing false information, destroying their ability to orient and attack or defend. Technology advancements in artificial intelligence (A.I.) already enable the use of decision-centric warfare that has yet to be fully leveraged. Combined with A2/AD strategies by adversaries, a new operational concept that leverages decision-centric warfare to mitigate the effects of an A2/AD strategy is necessary. Defense Advanced Research Project's Agency's (DARPA's) decision-centric warfare proposal, Mosaic Warfare, is designed to utilize components of manned and autonomous units, guided by human control, and can adapt to complex situations and dispute enemy centers of gravity, preventing further aggression (Clark et al., 2021).

Mosaic Warfare acknowledges the effect of an A2/AD environment and sets aside contemporary assumptions about communications that allow for centralized command and control of assets, instead leveraging local or *proximal* command and control of assets with a decentralized execution of a mission. Such an approach is different from the current Department of Defense doctrine for employing aircraft that relies on centralized planning and executing an Air Tasking Order (United States Joint Chiefs of Staff, 2021). In art, a painting or puzzle that is missing a piece is not whole; the absence visibly detracts from the work. In contrast, a mosaic is comprised of many individually-designed, simple tiles; their aggregation yields a work of art in the *grand mosaic*, even if one or more tiles are damaged or missing. Borrowing this paradigm, DARPA's strategic technology office (STO) compares contemporary doctrinal employment of aircraft to a jigsaw puzzle, wherein the loss of one piece damages the entire image (Magnuson, 2018). Mosaic Warfare mitigates this effect by leveraging functional tiles of warfighting platforms within a larger force package

(O'Donoughue et al., 2021; Sapaty, 2019). More specifically, Mosaic Warfare seeks to form functional tiles of manned and unmanned aircraft, wherein each tile has the needed capabilities to independently conduct part of an overall mission (Photonics Media, 2019). Moreover, these tiles are dynamic. As the proximity between aircraft changes – including with the addition or loss of aircraft – the number and composition of specific tiles evolve, retaining the functionality of both individual tiles and the grand mosaic. Thus, the grand mosaic within Mosaic Warfare is resilient to aircraft attrition, still capable of achieving operational success (DARPA News, 2017).

As an applied example, current air battle management in the US Air Force occurs via the E-3 Airborne Warning and Control System (AWACS) (Hoehn, 2022). The AWACS receives intelligence from intelligence, reconnaissance, and surveillance assets and, in a centralized manner, directs other aircraft to engage adversary aircraft and ground targets. If communications between the AWACS and other aircraft are disrupted, effective command and control is disrupted, as is situational awareness across the enterprise of assets. In contrast, air battle management within Mosaic Warfare occurs in a decentralized manner. A manned aircraft such as the F-35 Lightning II would communicate with and direct aircraft in relatively close proximity; the loss of communications across the enterprise of aircraft need not disrupt the effectiveness of local groups of aircraft, within which the hierarchical relationships of aircraft allow it to function as an effective tile.

Mosaic Warfare embraced an adaptive framework for decentralized, proximal command and control to leverage asset capabilities, maintain tactical momentum, and achieve military objectives in a communications-disrupted environment that prevents both centralized command and control. Moreover, the principles and technologies that develop a force design tailored to Mosaic Warfare will enable the U.S. to remain competitive against adversaries in future system warfare (Deptula et al., 2019). The

fundamentals of Mosaic Warfare entail the disaggregated information sharing related to radar, fire control, and missiles into smaller elements, rather than hosting them on a common platform for centralized command and control. The kill chain, which identifies the structure of attack into six phases (i.e., Find, Fix, Track, Target, Engage, Assess) (Tirpak, 2000), is implemented locally by smaller subsets of assets that function as tiles and, in aggregate, achieve an intended outcome – the mosaic. Mosaic Warfare's goal is to transform the operation into a kill web by limiting the number of critical nodes within the operational, decision-making network, allowing proximate, smaller groups of assets to remain effective in contested regions. Of note, asset membership in a tile is not fixed; membership evolves as missions change, disruptions to communications occur, or assets are rendered inoperative by an adversary (Deptula et al., 2019).

Mosaic Warfare will require more than the new technologies, it will also need a study of its operational concepts to examine the effectiveness of smaller, adaptive command and control structures to direct subsets of assets within missions. This research implements Mosaic Warfare for sets of semi-autonomous aircraft families utilizing concepts from the literature related to hierarchical local modeling and vehicle routing to plan and implement aerial strike missions. Missions of specific interest, each of which is carried out by a different type of aircraft, are detecting and observing targets with Intelligence, Surveillance, and Reconnaissance (ISR) aircraft; processing intelligence and direct action via command and control (C2) aircraft; and attacking targets using strike aircraft. Although multi-role aircraft are possible, this initial research considers single-role aircraft operating in a network-contested environment, so it is of interest to centrally plan their routes and adapt those plans in a decentralized manner when anticipated conditions change. If the C2 network becomes disrupted and feasible communication links are severed, disruption of information flow from

the sensor to strike asset must be addressed to maintain mission operation. The challenge will be to determine how to route simultaneous actors ensuring desired effects are achieved at targets while re-tasking and adapting rapidly as operational changes occur (Deptula et al., 2019).

2.1.1 Literature Review

The network dynamic of Mosaic Warfare is the culmination of various related networks and routing disciplines. The following section discusses previous works relating to location problems, primarily covering problems and hierarchical covering problems. It then explores the related literature on routing, both within multi-agent and multi-agent systems. Finally, grouping techniques to solve such problems are discussed, along with metrics to benchmark the performance of decentralized operations.

The foundational structure of the perceived Mosaic Warfare network model is a facility location problem because the purpose of grouping and moving assets is to get them close enough to *cover* demands (i.e., targets), both by sensors to positively identify the targets and strike aircraft to destroy them. Weber (1909) studied the central problem of location theory, which involves determining the optimal path on a plane to minimize the sum distance to n locations of interest. Generic facility location problems investigate locating assets such that demand is satisfied. As a convention common to several modeling frameworks, binary decision variables, x_{ij}, indicate whether a demand at location i has its demand fulfilled by a facility at location j, with an associated cost, c_{ij}. A binary variable y_j denotes the decision to emplace a facility at location j, at the cost of f_j, with a bound of the possible demand that can be serviced, u_j (Dimitri, 1998). Should the number of customers served be unbounded, allowing any number of demands to be served by a facility, the problem is an uncapacitated facility location problem (UFLP). Alternatively, a capacitated fa-

cility location problem (CFLP) is bounded by the amount of customer demand that a facility location can fulfill. Both UFLP and CLFP are Non-Deterministic Polynomial-hard (NP-hard) problems (Wu et al., 2006). Extensions of the classic facility location problem have included adding additional new locations to an already existing network by simply adding the new facilities with given interactions between themselves and the demand assets in a multi-facility model (Miehle, 1958), or allocating each demand node to a specific facility for service in a location-allocation model (Cooper, 1963). Location problems are often solved as p-median problems, a subclass of the minisum location models that focus on emplacing a fixed number of p facilities to minimize the weighted average of all distances in the system (Drezner and Hamacher, 2004).

A subset of facility location problems are covering problems, formalized when Hakimi (1965) investigated the minimum number of police officers necessary to distribute on a highway network such that no one would ever be more than d distance away from a policeman. Schilling (1993) categorizes covering models as either location set covering problems (LSCP) or maximal covering location problems (MCLP). The LSCP minimizes the number (or cost) of facilities necessary such that all demands are within a given distance of an emplaced facility. Alternatively, an MCLP attempts to provide as much coverage as possible, given a limited number of facilities available (Church and Murray, 2018). LSCP is appropriate where coverage is required, and MCLP is better suited for a limited set of assets. (Farahani et al., 2012). Applications of such modeling frameworks include fire station location (Badri et al., 1998), security camera placement (Yabuta and Kitazawa, 2008), cell phone coverage after a natural disaster (Eiselt and Marianov, 2012), and other variants (e.g., see (Farahani et al., 2012)).

The desired model to address the problem considered herein is a special case of the set covering model, where assets maintain a hierarchical relationship based on a

defined, established relationship, i.e., C2 assets must cover ISR and strike assets for information sharing and decision-making actions and, in turn, these assets must cover the target(s) being attacked. Hierarchical facility location problems (HFLP) address a multi-level network to determine the location of facilities and meet demand at the lower level of the hierarchy to minimize cost and maximize coverage. Farahani et al. (2014) details examples of models, classification, applications, and techniques of hierarchical facility location problems. Hierarchical networks are used in the education system to determine the number and locations of kindergartens, primary schools, and high schools in a district. Emergency medical service systems emplace and operate hierarchical facilities that administer different levels of medical treatment to the local area. Hierarchical facility location problems are deeply rooted in the healthcare field with studies to determine infirmaries, clinics, medical stations, and other health care specialists in the system. The waste management system seeks to properly place transfer stations, disposal centers, and landfill stations, which collectively manifest such relationships. Finally, hierarchical models in the production-distribution system usually exhibit different levels for factories, warehouses, and retail outlets, with associated demand for each location (Farahani et al., 2014).

In addition to covering targets, a Mosaic Warfare model must route assets to the demand locations. Dantzig and Ramser (1959) introduced some of the earliest routing problems, which identified the optimal routes for a fleet of vehicles to deliver supply to a set of customers. Whereas the Traveling Salesman Problem (TSP) is a routing problem where a salesman seeks to visit every node on a network and return to the starting node while minimizing distance, the Vehicle Routing Problem (VRP) considers multiple vehicles to service node-specific demands and with a carrying capacity for each asset. Routing problems fall into the general class of network optimization problems, the math programming formulation determines the complexity of the prob-

lem, which may be realized as the number of arcs and nodes (and possibly vehicles) increase (Anbuudayasankar et al., 2016). Karp explored the complexity of Hamiltonian circuit problems, a variant of the TSP where one visits each node exactly once before returning to the starting node (Karp, 1972). The TSP has been shown to be NP-complete (Lenstra and Kan, 1976), whereas the VRP is NP-hard (Karp and Papadimitriou, 1982).

The consideration of multiple agents routing simultaneously, each with its own set of constraints, parameters, and goals, imposes additional complexity to the standard VRP. The multiple vehicle routing problem (mVRP) consists of finding the optimal paths for m vehicles from a single depot (Nallusamy et al., 2010). The multi-depot vehicle routing problem further expands the VRP modeling framework, where vehicle distribution expands to origination from several locations (e.g., see Jayarathna et al. (2020)). The problem proposed herein studies three asset types (i.e., C2, sensor, and strike), each of which has a unique initial location that is almost akin to a depot, except the assets need not return to those points at the conclusion of the mission.

Because Mosaic Warfare entails the formation of tiles of assets into a functional group – and periodic realignment of assets between tiles – en route to targets, it is relevant to appreciate the need to leverage a grouping method. Methods such as the k-means clustering algorithm have been utilized to group cities to solve mVRPs via a decomposition of the problem to solve m single VRPs (Nallusamy et al., 2010). Location-and-routing problems, which explore the location selection of facilities and subsequent routing of vehicles to service demands, have utilized clustering techniques to similarly decompose and simplify the problem structure, informing high quality heuristics (Nadizadeh et al., 2011).

Formalized by Koutsoupias and Papadimitriou (Koutsoupias and Papadimitriou, 2009), the *Price of Anarchy* (POA) serves as an efficiency metric of a system with

decentralized decision-making, as compared to an "ideal" system with perfect information, assured communications, and centralized decision-making. Thus, POA is the measure of suboptimality introduced by the agents' self-interested behavior. The POA of a game is defined using a given notion of equilibrium (Nash equilibrium being popular in the literature) and an objective function, such as a summation of an agent's cost for utilizing a set of actions (Roughgarden, 2009). Defining actions at equilibrium as $\boldsymbol{x}$ and the set of actions that globally minimizes the objective function as $\boldsymbol{x}^*$, the respective objective function values are denoted as $f(\boldsymbol{x})$ and $f(\boldsymbol{x}^*)$. From these computations, the POA metric is calculated as $\frac{f(\boldsymbol{x})}{f(\boldsymbol{x}^*)}$.

Assumed is that $f(\boldsymbol{x}^*) > 0$, yielding a POA in $[1, \infty)$. For problems wherein $f(\boldsymbol{x}^*)$ can equal 0, POA is typically bounded artificially to be no less than 1. For a decentralized routing problem where the agents act in their interest (e.g., see (Frank, 1981)), a value close to 1 indicates relatively little additional cost (i.e., anarchy) is induced by the decentralized nature of the decision-making (Shoham and Leyton-Brown, 2008). A variety of applications apply the POA metric, such as network creations, where players create edges in a network to connect all parties while attempting to minimize the total cost of building the final network and total distance to all individuals participating (Fabrikant et al., 2003; Albers et al., 2014), similar to road construction connecting nearby towns and villages. Perakis and Roels (2007) utilized the POA to measure the efficiency of a decentralized supply chain that focuses on price-only contracts by looking at the profits from a fully coordinated network to one that is decentralized. Job scheduling has a history of leveraging a system's POA to measure the efficiency of scheduling policies (Czumaj et al., 2002; Ye et al., 2021). Other recent works from literature utilized the POA for location and routing models, including multi-agent unmanned aerial vehicle (UAV) systems (Thakoor et al., 2019). Although this study is not examining the efficiency loss on a network due

to self-interested agents in the system, the concept of a POA is useful to quantify the relative inefficiencies of implementing Mosaic Warfare via the proposed solution techniques and under different user-defined algorithmic parameters.

In its essence, Mosaic Warfare is a multi-agent control problem. A multi-agent system considers an environment, a set of agents with an assembly of relations, operators, and taskings (Ferber and Weiss, 1999). However, it is unlike traditional problems from that stream of literature because of the hierarchical relationships between assets; although there is not centralized command and control, it is not completely decentralized. Instead, Mosaic Warfare relies on *regionally centralized* command and control, which occurs via C2 assets directing proximal assets within their respective tiles. As such, this research builds upon the most relevant aspects within the literature pertaining to grouping, routing, and covering problems to implement and assess this operational concept.

2.1.2 Statement of Contributions

This research makes the following two contributions. First, it develops and assesses an algorithmic procedure to implement the Mosaic Warfare operational concept via an iterative combination of asset grouping, asset routing, and demand coverage. Complementary to this procedure are POA-like efficiency metrics to assess the relative cost of implementing Mosaic Warfare, as well as user-defined parameters relating to the frequency with which tiles may be reformed and individual asset agency within a tile. Second, for a set of representative scenarios within a designed experiment, this research examines the effect of selected problem and algorithmic parameters on solution efficiency via computational experiments.

The rest of this paper is organized as follows. Section 2.2 details the proposed Hierarchical Asset Tiling and Routing Heuristic (HATRH) with specific discussion

regarding critical aspects of the HATRH, including how assets are grouped to ensure Mosaic Warfare tiles are functional, how tiles vis-á-vis individual assets are moved, and how-and-when targets may be covered. Section 2.3 introduces the test design of parametric and algorithmic features as well as scenarios representing different spatial relationships between initial asset and target locations, after which it presents and analyzes the testing results. Section 2.4 concludes with a summary of garnered insights and proposes directions in which to extend this research.

2.2 Solution Methodology

This research applies Mosaic Warfare to move friendly aircraft through a region wherein inter-asset communications are limited, to attacking a set of stationary ground targets. Within this environment that precludes centralized command and control of all assets but allows local command and control, the process of attacking a target requires a functional tile or group. A functional tile requires at least one each of a C2, sensor, and strike aircraft. To attack a target, at least one sensor aircraft within the tile must be close enough to the target to positively identify it; at least one strike aircraft must be close enough to the target to destroy it; and a C2 aircraft must be close enough to both the sensor and strike aircraft to receive the positive identification of the target, make a decision regarding how to destroy the target (e.g., what type of munition), and direct the strike aircraft to attack. Because decision-making is central to a tile's function, an application of Mosaic Warfare must form (and iteratively reform) tiles around functioning C2 aircraft as a mission proceeds.

The algorithmic modeling implementation of selected aspects of Mosaic Warfare proposed explores the routing of multiple single-role assets that form functional tiles when grouped. Such routing procedures will explore the dynamic between focusing on group and individual asset agency, where assets are encouraged to center priority

on group movement versus individual movement capabilities, respectively. As such, the methodology presented relaxes limitations such as operational fuel and armament capabilities, probability of kill modeling, and adversary opposition. Critical functions in a mosaic system require identifying and routing potential assets along multiple and simultaneous kill paths, ensuring desired effects are delivered, then re-tasking assets appropriately (Deptula et al., 2019). As early quantitative analysis exploring Mosaic Warfare in the literature, this algorithmic implementation of Mosaic Warfare makes some simplifying assumptions to focus analysis on the relative efficiencies for selected performance metrics.

As presented in Algorithm 1, the HATRH algorithm iteratively applies three procedures until there are no active targets (i.e., all targets destroyed). In Line 2, HATRH invokes a *Grouping Algorithm* to form (or reform) tiles of proximal assets to ensure each eligible C2 aircraft has at least one sensor and one strike within its tile. In Line 3, HATRH applies a *Tile Movement* Algorithm to myopically advance all assets within each tile towards its nearest target without coordination between tiles. In Line 4, HCHR invokes an *Individual Movement Algorithm* to allow a bounded degree of individual sensor and strike aircraft autonomy relative to their tile's C2 aircraft. Lines 5-9 check each active target against all tiles to determine if a sensor and strike asset in the same functional tile are both within range. If at least one tile meets these criteria, Line 7 removes the target from the active target list.

Before detailing each of the three subroutines invoked by HATRH, it is important to define some notation commonly used by each of these algorithms and initially characterize two user-defined HATRH parameters. Hereafter, the discussion uses $C2$, S, St, and T to respectively denote the sets of C2 assets, sensor assets, strike assets, and active targets, and it indexes individual assets within each set (e.g., $C2_i$ references the i^{th} C2 asset). The discussion will also leverage the notation ϕ and

Algorithm 1 Hierarchical Asset Tiling and Routing Heuristic

```
1: while number of active targets > 0 do
2:     Perform Grouping Algorithm                    ▷ Section 2.2.1, Algorithm 2
3:     Perform Tile Movement Algorithm               ▷ Section 2.2.2, Algorithm 3
4:     Perform Individual Movement Algorithm         ▷ Section 2.2.3, Algorithm 4
5:     for all active targets do
6:         if S and St assets within same tile are within range then
7:             remove that target from the list of active targets
8:         end if
9:     end for
10: end while
```

λ. The parameter ϕ is a *tile movement parameter* that will be discussed in detail in Section 2.2.2. In practice, for a set of tiles moving toward their self-designated targets, ϕ limits how close each tile can move towards its target before HATRH reevaluates the tiling of assets. The parameter λ is an *asset movement parameter* that will be detailed in Section 2.2.3. In general terms, it apportions each asset's allowable movement within an iteration of HATRH between movement directed for all assets within a tile and asset-specific movement (i.e., movement closer to the tile's C2 asset if required for functional communication or movement relatively closer to the designated target).

2.2.1 Grouping Algorithm

The Grouping Algorithm forms functional tiles of sensor and strike assets around C2 assets because these assets are central to the targeting process. Within each HATRH iteration, the Grouping Algorithm may affirm existing asset tiles or realign assets between tiles. The latter outcome is more likely when tiles are in close proximity to each other or when assets become inoperable.

Of relevance when forming tiles is the relative proximity of assets. If a C2 asset is close enough to at least one strike and one sensor asset to form a functional tile as it moves towards a target within the current iteration, that C2 asset is considered to

be *eligible* to have sensor and strike assets assigned to it within a tile. For expository convenience, we use the term *proximal* to indicate that a sensor or strike asset is "close enough" within this context. For the remaining eligible C2 assets, the Grouping Algorithm first assigns at least one proximal sensor asset and one proximal strike asset. It then assigns the remaining sensor and strike assets to tiles. It assigns any C2 assets not eligible for grouping to the nearest eligible C2 assets to follow their movement. The Grouping Algorithm implements a default assignment procedure for the extreme case when no sensor or strike assets are proximal to any C2 assets. Algorithm 2 presents the Grouping Algorithm used within the HATRH.

Algorithm 2 Grouping Algorithm

1: **for all** $i \in C2$ **do**
2: Remove asset $C2_i$ from consideration if no sensor assets *or* no strike assets are proximal
3: **end for**
4: **if** at least one C2 asset is eligible for grouping **then**
5: **for all** $i \in C2$ for eligible C2 assets **do**
6: **if** nearest ungrouped asset is proximal to $C2_i$ **then**
7: Assign it to $C2_i$
8: **else**
9: Reassign nearest proximal asset from $C2_j$ to $C2_i$ (where $j < i$)
10: **end if**
11: **end for**
12: Assign closest remaining sensor and strike assets to closest eligible C2 asset
13: **if** every eligible C2 asset does *not* have an assigned, proximal sensor and strike asset **then**
14: Remove the highest-indexed eligible C2 asset from consideration
15: Unassign all sensor and strike assets
16: Proceed to Line 4
17: **end if**
18: Assign all remaining assets to the closest, eligible C2 assets
19: **else**
20: Assign all assets to $C2_1$ as a single tile
21: **end if**

Lines 1-3 limits consideration for grouping to only C2 assets having proximal sensor and strike assets. Lines 4-17 assign the sensor and strike assets to these eligible

C2 assets within a tile. Lines 4-11 iteratively apply a greedy heuristic to assign the nearest asset of each type to a C2 asset. In ascending order by the eligible C2 asset index, Lines 5-11 assign the nearest, proximal asset of each type to $C2_i$. If no such asset exists, the procedure reassigns an eligible proximal asset from a lower-indexed C2 asset, removing it from the previously assigned tile. Line 12 assigns the remaining sensor and strike assets to the closest, eligible C2 assets, which may, in turn, resolve any non-functional tile issues. If it does not, Lines 13-17 reduce the set of eligible C2 assets as a means to resolve the conflict and reruns the assignment of sensor and strike assets by returning to Line 3. If all eligible C2 assets have a functioning tile with proximal sensor and strike assets, Line 18 assigns all remaining C2 assets to their nearest tiles. In the extreme case where no functional tiles can be formed, Line 20 assigns all assets to a single tile, and the HATRH procedure continues.

2.2.2 Tile Movement Algorithm

The Tile Movement Algorithm determines the synchronous movement direction and distance traveled by the formed tiles. More specifically, the Tile Movement Algorithm determines the respective targets the tiles move towards and how far each tile moves in their respective directions.

The Tile Movement Algorithm manages each tile with respect to its centroid, a cardinality weighted average of all asset locations within the tile. After identifying the centroid, it assigns each tile to its closest target or *target of interest* (TOI), informing a centroid-to-TOI vector. This algorithm moves all assets within a tile from their current positions along the same vector.

The distance traveled by each asset within a tile is equal to the minimum of a λ-informed tile movement distance and a ϕ-informed bound on the relative closure of the tile to its TOI. More specifically, if an asset can move a distance of at most

q units during an iteration, the parameter λ indicates the percentage of that movement reserved for individual asset movement, independent of tile-directed movement. Lower λ-values reduce individual asset autonomy and tend the preserve the relative positions of tile assets with respect to each other. By comparison, the parameter ϕ is the maximum proportion of the centroid-to-TOI distance the tile is allowed to travel within an iteration. Lower ϕ-values will limit tile movement and require more iterations of the HATRH, compelling more revisits of asset grouping decisions. Within Section 3, a designed computational testing experiment examines the effect of both λ- and ϕ-values on solution quality with respect to the metrics presented in Section 2.5.

Preliminary testing did identify a pathological tile movement case that the algorithm addresses. It is possible to have a negligible centroid-to-TOI distance, e.g., for a tile with assets spread radially around its closest target but not within range to destroy it. For such a case, the standard procedure within the Tile Movement Algorithm would not compel the assets to converge on the target because the centroid-to-TOI distance is small. This algorithm identifies such a case based on a small centroid-to-TOI distance and, instead of directing the tile's assets to move along a negligibly small vector (e.g., 1% of the tile movement limit), it directs each asset within the tile to move towards its centroid, akin to the procedure within the Nelder Mead Algorithm that shrinks its simplex (Nelder and Mead, 1965). Algorithm 3 presents the Tile Movement Algorithm used within the HATRH.

Lines 1 and 2 initialize the algorithm by respectively calculating the maximum distance any tile can move within an HATRH iteration without regard to their centroid-to-TOI proximity and a user-defined tolerance for identifying the aforementioned pathological tile-to-TOI case. Preliminary testing indicated that one percent of the tile movement limit performed well for the instances examined in this study. How-

Algorithm 3 Tile Movement Algorithm

1: Set tile movement limit = min{$(1-\lambda)$ S/St movement limit, C2 movement limit}
2: Set pathological movement tolerance $\epsilon = 0.01$ tile movement limit
3: **for all** $i \in$ tiles **do**
4: Calculate centroid for tile$_i$
5: Identify TOI$_i$ and centroid-to-TOI$_i$ vector v_i
6: Calculate movement length for $l_i = \min\{\text{tile}movelimit, \phi||v_i||\}$
7: Move all assets in tile i a distance of l_i in direction v_i
8: **if** $l_i < \epsilon$ **then**
9: **for all** assets j in tile i **do**
10: Identify asset-centroid vector u_j
11: Move asset a distance of min {asset movement limit $- l_i, \phi||u_j||$ } towards the tile centroid
12: **end for**
13: **end if**
14: **end for**

ever, this parameter should be adjusted as necessary to accommodate the scale of an instance.

Lines 3-7 perform tile movement. Line 4 and 5, respectively, determine each tile's centroid and each tile's TOI and movement vector v_i. Line 6 calculates the movement length for each tile, and Line 7 moves each of the assets within a tile at the same distance l_i, in the same direction v_i.

Subsequently, Line 9 identifies the existence of pathological tile-to-TOI movement. Line 10 identifies each asset to centroid vector u_j and appropriately moves all assets within the tile directly towards the centroid at a distance bounded by Line 11 along the determined vector.

2.2.3 Individual Asset Movement Algorithm

Applied after the Tile Movement Algorithm, the Individual Asset Movement Algorithm accounts for a degree of individual asset autonomy informed by the parameter λ, moving assets within each C2-oriented tile. This algorithm ensures that asset-specific movement within an iteration (i.e., λ) seeks to address two priorities. First,

it ensures any sensor and strike assets assigned to the tile but not proximal to their closest C2 asset within said tile, denoted $C2_k$, move within – or close to within – the communication range of the this C2 asset. Second, it seeks to move sensor and strike assets closer to the TOI without moving outside the communication range of the assigned C2. As the C2 assets serve as the foundational tiling element, no further movement will occur during this portion of the HATRH as their role shifts from routing to ensuring proper relaying of communication commands to subsidiary assets in the hierarchical chain.

Given the asset-specific movement allowed by λ, this algorithm is relatively intuitive for its first priority. After identifying any assets not within communication range of the nearest C2 within the tile, it moves them the minimum distance to be within range or, if that is too far within an HATRH iteration, moves them as close as possible to that point.

The Individual Asset Movement Algorithm's second priority applies to assets within communications range of the $C2_k$ asset. Given the relative locations of the $C2_k$ asset within a tile, a TOI, and a sensor or strike asset (denoted S_i/St_i) within communications range of the C2 asset, Figure 1 depicts the geometry pertaining to identifying the ideal asset location α and the corresponding distance β. Note that α is as close as possible to the TOI along the S_i/St_i-to-TOI vector while remaining within communications range of the $C2_k$ asset. The Individual Asset Movement Algorithm will move the asset to point α or, if the λ-informed individual asset movement precludes it, move it as close to α as possible, unless the distance to the TOI minus the asset's defined range is less.

Simple trigonometric identities inform the calculation of α, as depicted in Figure 1. Via Equation (1), θ_1 is computed using the Law of Cosines. Equations (2) and (3) respectively compute θ_2 and θ_3 using the Law of Sines and the property that the sum

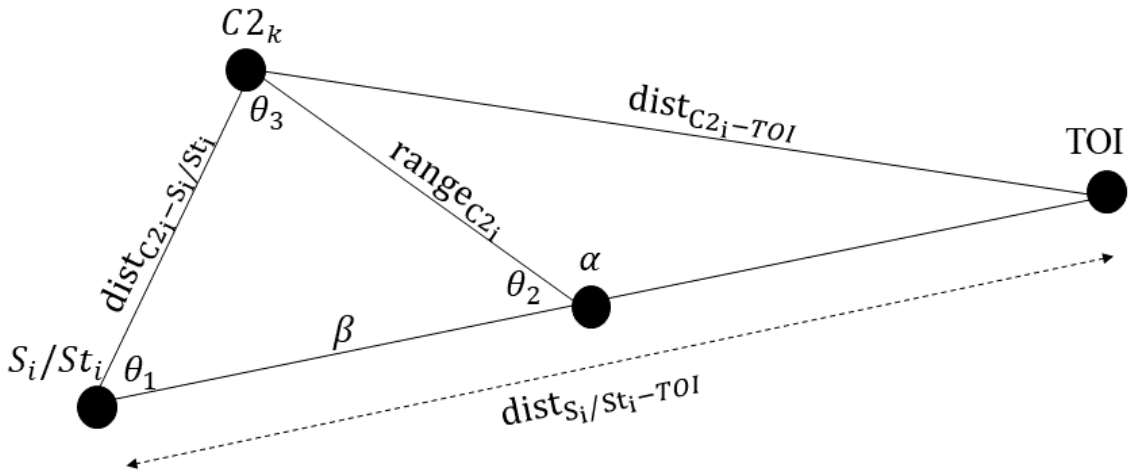


Figure 1. Guide to Finding Optimal Movement Point

of a triangle's interior angles equals π radians. Finally, Equation (4) computes β via the Law of Sines. Algorithm 4 presents the HATRH's Individual Asset Movement Algorithm.

$$\theta_1 = cos^{-1}\left(\frac{\text{dist}^2_{\text{C2-S/St}} + \text{dist}^2_{\text{S/St-TOI}} - \text{dist}^2_{\text{C2-TOI}}}{2(\text{dist}_{\text{C2-S/St}})(\text{dist}_{\text{S/St-TOI}})}\right) \tag{1}$$

$$\theta_2 = sin^{-1}\left(\frac{sin(\theta_1)(\text{dist}^2_{\text{C2-S/St}})}{\text{range}_{\text{C2}}}\right) \tag{2}$$

$$\theta_3 = \pi - \theta_1 - \theta_2 \tag{3}$$

$$\beta = \frac{sin(\theta_3)range_{C2}}{sin(\theta_1)} \tag{4}$$

Lines 1-13 examine each tile formed during the current HATRH iteration, and Lines 2-12 direct the movement of sensor and strike assets within a given tile. Line 3 determines the closet C2 asset within the assigned tile the individual assets will attempt proximal movement. For an asset beyond possible movement range to the

Algorithm 4 Individual Asset Movement Algorithm

1: **for all** tiles **do**
2: **for all** sensor and strike assets in a tile **do**
3: Let $C2_k$ be the closest C2 asset in the tile to the sensor (or strike) asset
4: **if** asset-to-$C2_k$ distance $\geq$ (asset movement limit + $range_{C2}$) **then**
5: Move asset as close to $C2_k$ as adjustment limit allows
6: **else**
7: Let m_j=minimal distance an asset j move to be within range of $C2_k$
8: Move m_j towards $C2_k$
9: Calculate β via Equations 1-4
10: Move asset a distance of $\min\{\beta, \text{asset movement limit} - m_j, dist_{S_i/St_i\text{-}TOI} - range_{S/St}\}$ towards the TOI
11: **end if**
12: **end for**
13: **end for**

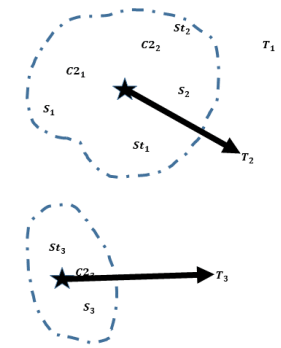
closest C2 within the tile, Lines 4-5 move it as close as possible. Otherwise the asset moves the minimal distance necessary to be within range of the closest determined C2 asset (Lines 7 and 8). Lines 9 and 10 move the assets as close as possible to point α or the TOI as necessary.

2.2.4 Illustrative Application of an HATRH Iteration

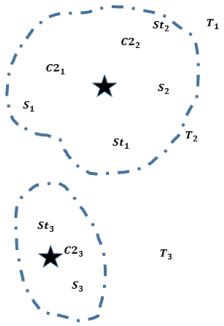
The dynamics and interplay of the algorithms within the HATRH as applied to the different types of assets merit illustration via an example. Figure 2 depicts a single iteration of the HATRH applied to a set of assets. This instance has three assets of each type: C2 ($C2_1$, $C2_2$, $C2_3$), sensors (S_1, S_2, S_3), strike assets (St_1, St_2, St_3), and targets (T_1, T_2, T_3).

Within the HATRH iteration, Figure 2a depicts the results of applying the Grouping Algorithm and the initial steps of the Tile Movement Algorithm. $C2_1$ does not have both a sensor and strike asset proximal to it, so it cannot form a functional tile. The algorithm initially assigns S_2 and St_2 to $C2_2$, and S_3 and St_3 to $C2_3$, after which is assigns the remaining assets (i.e., $C2_2$, S_1, and St_1) to the nearest C2 asset (i.e., $C2_2$

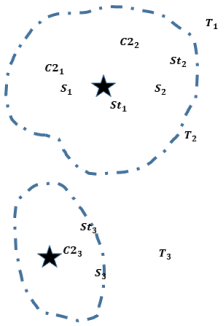
for each such asset). Also depicted in Figure 2(a) from the Tile Movement Algorithm is the identification of each tile's centroid, TOI, and centroid-to-TOI vector.



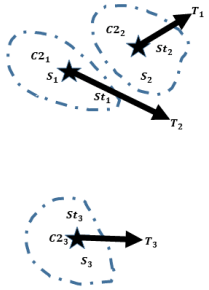
(a) Grouping and preliminary Tile Movement calculations



(b) Tile Movement



(c) Individual Asset Movement, HATRH iteration s



(d) Grouping and preliminary Tile Movement calculations, subsequent HATRH iteration

Figure 2. HATRH Single Iteration Walkthrough

Figure 2b shows the result of the Tile Movement Algorithm, with all tile movements for this instance bounded by $\phi = 0.25$. Figure 2c presents the outcome of the Individual Asset Movement Algorithm. The algorithm moved assets S_1 and St_1 as close to being in communications range of $C2_2$ as permitted by λ, and it moved the remaining sensors and strike assets as close to their respective α-points as allowed, to move towards their TOIs. To conclude the HATRH iteration, no targets are within range of both a sensor and a strike asset within a functional tile, so all targets remain active.

For the subsequent HATRH iteration, Figure 2d shows the regrouping of assets.

Each C2 asset has proximal sensor and strike assets, so the Grouping Algorithm forms three functional tiles. During preliminary calculations, the Tile Movement Algorithm identifies and Figure 2d depicts a TOI and a tile-to-TOI vector for each tile.

This illustrative instance of an HATRH exhibited several beneficial outcomes of HATRH. The subsequent iteration formed the maximal number of possible functional tiles, and there was a one-to-one assignment of tiles to targets. However, such outcomes are not assured. While implementing the distributed, dynamic command-and-control precepts of Mosaic Warfare, the HATRH may route assets inefficiently, or it may assign multiple assets to destroy a single target, a realistic outcome of Mosaic Warfare. Within this context, it is of interest to characterize and quantify the relative inefficiencies of Mosaic Warfare relative to an operational environment where communications are not limited, and Mosaic Warfare is not necessary.

2.2.5 HATRH Evaluation Metrics

Mosaic Warfare exhibits the potential for two critical inefficiencies: asset routing and asset-to-target assignment. Whereas the former inefficiency can waste fuel and time, the latter can waste munitions. Moreover, the levels of inefficiency may be affected by instance-specific characteristics such as the initial, relative disposition of assets and targets, and the congestion of assets within a region. The user-defined parameters λ and ϕ may also affect these outcomes. Before examining such effects, it is important to formalize a POA-like metric to quantify each of these potential inefficiencies.

The first POA-like[1] metric involves the number of engagements during an operation. During the execution of the HATRH, it is feasible for two tiles to arrive at and destroy a TOI within a single iteration. This outcome is reasonable for Mosaic

[1]We use the term *POA-like* rather than *POA* because Mosaic Warfare does not represent complete anarchy; it manifests a lower level of chaos via localized rather than centralized command and control.

Warfare, which does not assume that groups of assets (i.e., tiles for the HATRH) can necessarily communicate, so each tile may expend munitions to destroy the same target. In contrast, a perfectly efficient operation would expend only one munition per target. Thus, the ratio between the number of tile-to-target engagements by the HATRH to the number of targets is our munitions efficiency metric, or POA_{me}.

The second POA-like metric measures route inefficiency. If an operational environment allowed for centralized communication with complete information about the set of targets (i.e., no sensors need positively identify the targets, and no C2 assets require routing), the most efficient routing of assets would minimize the average distance traveled by the strike assets, the assets that launch the munitions to destroy the targets. By comparison, applying the HATRH to an instance will yield an average distance traveled by strike assets to destroy the set of targets that may not be efficient. Herein, we adopt the average strike distance ratio, POA_{asd} as a second performance metric for Mosaic Warfare, computing it as the ratio between the actual and optimal average distance traveled by strike assets to destroy the set of targets.

The efficient, average strike distance can be identified via the optimal solution to a math program that uses the following sets, parameters, and decision variables.

Sets

- $N = \{1, ..., n\}$: The set of nodes representing the initial location of all strike and target assets in the operation, indexed alternatively on either i or j.
  - $N_t \subset N$ is the subset of nodes corresponding to targets.
- A: The set of directed arcs indexed by (i, j), for all $i \in N$ and $j \in N_t$.
- $G(N, A)$: the directed network.
- K: the set of strike assets, indexed on k.

Parameters

- d_{ij}: length of arc (i, j)
- b_{ik}: binary parameter equal to 1 if strike asset k is initially located at node i, and 0 otherwise.

Decision Variables

- w_{ijk}: a binary variable equal to 1 if strike asset k traverses arc (i, j), and 0 otherwise.
- u_{ik}: non-negative variable specific to each node i and strike asset k, used within the formulation to implement Miller-Tucker-Zemlin (MTZ) subtour elimination constraints (Miller et al., 1960).

With the aforementioned notion, we formulate Problem **P** to identify the minimal distance routing of strike assets to visit all target locations.

$$\mathbf{P}: \min_{\mathbf{w},\mathbf{u}} \frac{1}{|K|} \sum_{(i,j)\in A} \sum_{k\in K} d_{ij} w_{ijk} \tag{5}$$

$$\text{s.t.} \sum_{j:(i,j)\in A} w_{ijk} - \sum_{j:(j,i)\in A} w_{jik} \leq b_{ik}, \ \forall\ i \in N, k \in K, \tag{6}$$

$$\sum_{i:(i,j)\in A} \sum_{k\in K} w_{ijk} \geq 1, \ \forall\ j \in N_t, \tag{7}$$

$$u_{ik} - u_{jk} + M w_{ijk} \leq M - 1, \ \forall\ (i, j) \in A, k \in K, \tag{8}$$

$$w_{ijk} \in \{0, 1\}, \ \forall\ i \in N, j \in N, k \in K, \tag{9}$$

$$u_{ik} \geq 0, \ \forall\ i \in N, k \in K. \tag{10}$$

The formulation seeks to minimize the average distance traveled by strike assets (5). Equation (6) enforces conservation of flow for the movement of strike assets. Equation (7) ensures that each target is visited by at least one strike asset, and Equation (8) applies the MTZ subtour elimination constraints. It suffices to set $M = |N_t|$. Finally, Equations (9) and (10) respectively enforce appropriate restrictions on the decision variables.

As a caveat, we recognize that certain efficient performances may not be attainable, or a decision-maker may not want to attain them. For example, an asset routing solution that minimizes POA_{asd} might only use one strike asset, forgoing the use of all others and extending the duration of a mission while minimizing the average strike asset distance. Alternatively, a solution that minimizes POA_{me} provides no redundancy when destroying targets, assuming that every munition will work perfectly and destroy a target with certainty. However, each of these POA-like metrics is suitable, identifiable, and ultimately useful to benchmark the performance of the HATRH algorithm to implement Mosaic Warfare for instances of the underlying problem.

2.3 Testing, Results, and Analysis

It is relevant to analytically test the effect of both user-defined algorithmic parameters and selected problem features on the efficacy of the HATRH algorithm vis-à-vis the POA metrics set forth in Section 2.2.5. The algorithmic parameters of interest are the tile movement parameter ϕ and the individual movement influence parameter λ. Problem features examined herein are the initial, relative dispositions of the set of assets with respect to the set of targets and the relative congestion of assets and targets within the area of interest.

We examine instances for a representative application of the motivating problem for testing. All initial asset and target locations instantiate within a 150 km $\times$ 150 km geographic region. Testing assumes movements, inter-asset communications, and targets are coplanar, so the depiction of results directly illustrates inter-asset distances. Informed values for the aircraft range and travel limitation parameters for C2, sensor, and strike aircraft within each iteration come from open-source, unclassified performance characteristics for the E-3 Sentry (U.S. Air Force, 2015), RQ-4 Global Hawk (U.S. Air Force, 2014), and F-16 Fighting Falcon (U.S. Air Force, 2021). More specif-

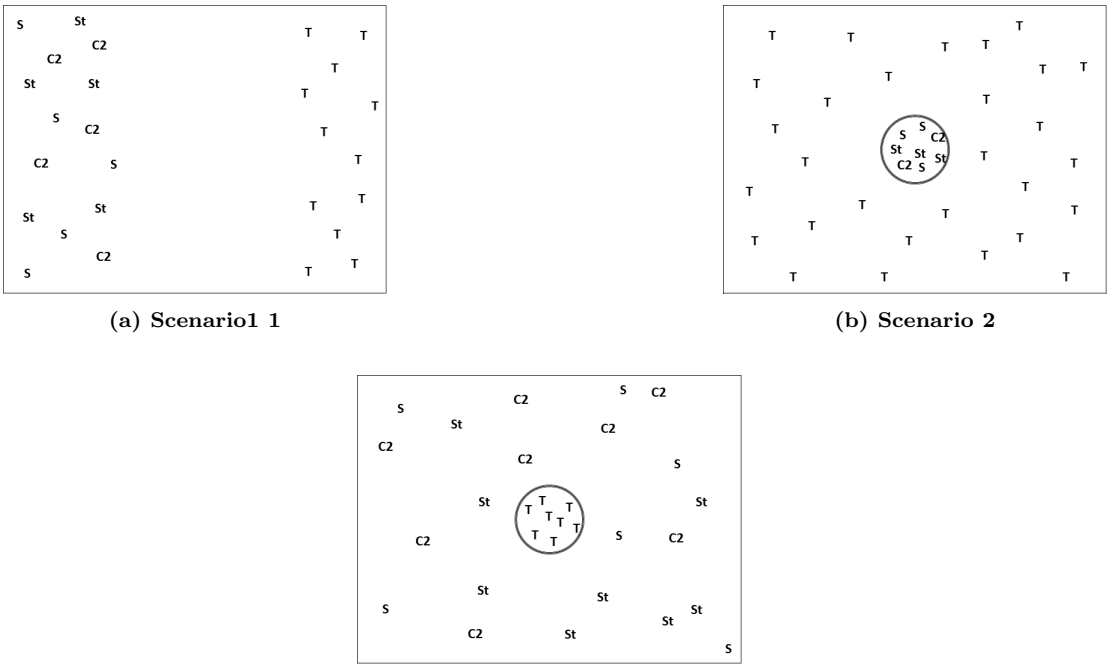
ically, the range for the C2 and sensor aircraft reflect the respective aircraft's communication transmission range (Jane's, 2021). In contrast, the strike asset's range is assumed to be that of the Joint Direct Attack Munition (U.S. Air Force, 2017), which is consistent with typical F-16 armament (Jane's, 2021). A logarithmic transformation and subsequent uniform, linear scaling of all asset parameters ensured instances scale to the 150 km $\times$ 150 km engagement region.

With respect to HATRH algorithmic parameters, a designed computational testing experiment examined three values of the tile movement parameter ϕ and 19 values of the asset movement parameter λ. We considered $\phi \in \{0.25, 0.51, 0.75\}$, with the middle value slightly above 0.5 to guarantee respective tile convergence on TOIs. Tested values of $\lambda \in \{0.05, 0.10, ..., 0.95\}$ represent a large range of relative movement autonomy reserved for individual asset movement.

Testing considered relative asset congestion feature as follows. Each instance was generated for a specified total number of assets (including targets). The affixed relative numbers of targets, sensors, strike assets, and C2 assets at proportions of 0.5, 0.2, 0.2, and 0.1 of the total number of assets. Testing considered 10, 30, 50, and 70 total assets initially located within the engagement area.

Of course, the relative disposition of the sets of assets and targets is another important problem feature, as it characterizes an engagement scenario. Testing herein considered three conceptually-motivated scenarios. Figure 3 illustrates an example of initial asset location for sensors (S), strike assets (St), C2 assets (C2), and targets (T) for Scenarios 1-3. Scenario 1 represents a relatively linear movement of assets toward targets, as may occur during an incursion into adversary-controlled territory. The movement is assumed to occur from left-to-right (without loss of generality) within the engagement area, with initial locations of sensors, strike aircraft, and C2 aircraft dispersed randomly within the leftmost 30% of the region and targets dispersed like-

wise in the rightmost 30%. Scenario 2 represents a defensive encirclement, as may occur when friendly assets begin an engagement in a relatively small region in the center of the engagement area, moving to destroy targets that encircle it. In contrast, Scenario 3 represents an offensive encirclement; wherein friendly assets initially encircle a set of targets in the center of the engagement area.



(a) Scenario1 1

(b) Scenario 2

(c) Scenario 3

Figure 3. Illustrative depiction of Scenarios 1-3 for relative disposition of initial asset and target locations

Testing applied the HRCH algorithm to solve 30 instances for each of the 684 combinations of algorithmic parameters and problem features. All testing was performed using Python (Version 3.9.7) to implement the HATRH on an Intel(R) Core(TM) i7-10875H CPU @2.30GHz with 128 GB of RAM on a 64-bit operating system. The optimal routing of strike assets to inform the calculation of average performance metric POA_{asd} was identified via the GurobyPi package to invoke the commercial solver Gurobi (Version 9.5.1) with termination criteria of 30 minutes of computational effort

and a 0.0001% relative optimality gap [2].

When finding the solution to Problem P to compute POA_{asd}, Gurobi was able to identify an optimal routing solution for all but the largest instances for a subset of scenarios. For such cases, subsequent discussions detail when that occurred, why it occurred, and how it may affect the corresponding assessments of Mosaic Warfare's relative effectiveness.

Because the scenarios are specific to conceptually-motivated engagements, the following discussions are scenario-specific. Sections 3.1-3.3 respectively present and discuss the results and insights for testing on 6840 instances each of Scenarios 1-3.

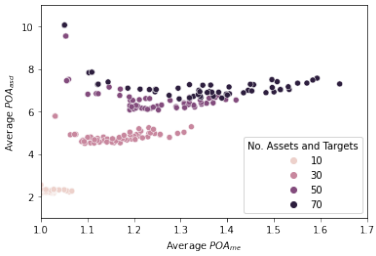
2.3.1 Testing Results for Scenario 1

Figure 4 presents the testing results for Scenario 1, with each of three subfigures depicting the average performance metrics POA_{me} and POA_{asd} on the respective horizontal and vertical axes for 30 instances of each of the 228 parameter and feature combinations. The respective subfigures differentiate the instances by problem feature or an HRCH parameter, each with the legend inset. Figures 4a, 4b, and 4c respectively illustrate the relative differences in instance performance due to relative asset congestion (i.e., total assets and targets), tile movement parameter (ϕ), and asset movement parameter (λ).

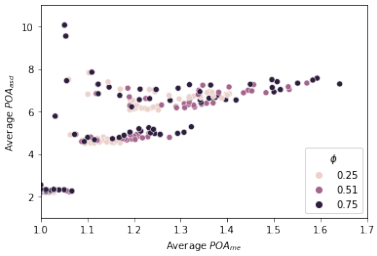
Of note, when calculating POA_{asd}, Gurobi identified the optimal solution to Problem P for instances having either 10 or 30 total assets-and-targets. For relative asset congestions having 50 and 70 players, Gurobi found feasible solutions with respective, average optimality gaps of 11.4% and 28.7%. Such feasible solutions to benchmark Mosaic Warfare performance may be optimal, but they are not assuredly so. This possibility indicates the results in Figure 4a are lower bounds on MOA_{asd} for 50 and

[2] An electronic copy of the Python code used to implement HATRH can be found at the following GutHub link: https://github.com/donnelsd/HATRH

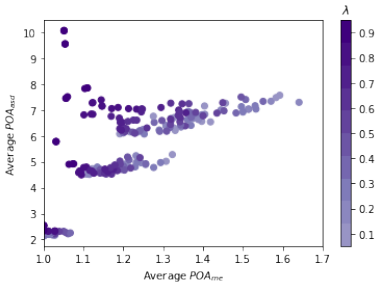
70 total assets-and-targets; the relative performance of Mosaic Warfare may be worse.



(a) Differentiated by relative asset congestion



(b) Differentiated by tile movement parameter (ϕ)



(c) Differentiated by asset movement parameter (λ)

Figure 4. HATRH-implemented Mosaic Warfare performance for Scenario 1, over 30 instances of each of 228 combinations of relative asset congestion, ϕ, and λ

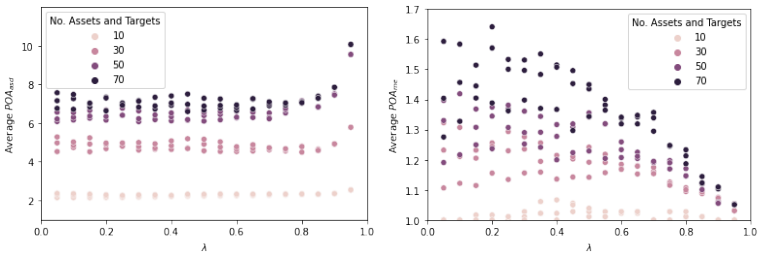
Nevertheless, the results exhibit compelling trends for specific HATRH algorithmic parameters or problem features. Foremost, Figure 4a shows that an increase in asset congestion portends more inefficient routing of assets (i.e., POA_{asd}). Three aspects of Mosaic Warfare allow for this inefficiency. First, strike assets are grouped and may be limited in movement by their assigned tile. Second, strike assets may iteratively regroup between different C2 assets. Third, tiles myopically attack the nearest TOI. The likelihood of each of these three potential inefficiencies increases with greater asset congestion, as is manifest in Figure 4a. Such a trend is not as starkly evident for munitions use (i.e., POA_{me}), but the minimum, average, and maximum values for average POA_{me} each increase as asset congestion increases. This type of inefficiency occurs when multiple tiles engage the same target(s) within an HATRH iteration, which is also more likely with higher relative asset congestion.

Specific trends are more subtle within Figure 4b. As the ϕ increases, the maximum values for average POA_{me} often increase, implying that larger movements between reregrouping can yield greater inefficiencies for asset movements. However, such an outcome does not hold, on average. Moreover, for a given level of relative asset congestion, higher values of ϕ portend slight increases in average POA_{asd}. However, the relative effectiveness of HATRH-implemented Mosaic Warfare is not significantly affected by ϕ for Scenario 1. This result indicates that more frequent asset regrouping does not always improve Mosaic Warfare's relative effectiveness. For instances akin to Scenario 1, an initial grouping of assets may suffice, and one might forgo a regrouping of assets in the absence of a specific need to do so (e.g., if an adversary destroys friendly assets).

In contrast, Figure 4c shows a readily discernible effect of λ on Mosaic Warfare performance via the HATRH. Its greatest impact is on POA_{me}-values; as λ increases for a given level of relative asset congestion, POA_{me} generally decreases (i.e., improves), whereas POA_{asd} initially decreases before increasing gradually.

Figure 5 explores this effect of λ in greater detail. Figures 5a and 5b respectively display POA_{asd}- and POA_{me}-values for increasing values of $\lambda \in \{0.05, 0.1, ...0.95\}$ and, at each λ-values, for 30 instances each of the 12 combinations of ϕ and relative asset congestion settings. Both figures differentiate the instance performances by the relative asset congestion, again with legends inset.

Recall that higher $\lambda-$values indicate the Tile Movement Algorithm controls a lesser proportion of individual asset movement limits. Higher λ-values reserve movement capability for use by the Individual Asset Movement Algorithm that affords greater agency to sensor and strike assets to move towards a TOI. Within Figure 5a, as λ increases beyond 0.80, the $POA_{asd}-$values increase for a given level of relative asset congestion, more so for higher congestion levels. This result implies that indi-



(a) Effect of λ on POA_{asd} **(b) Effect of λ on POA_{me}**

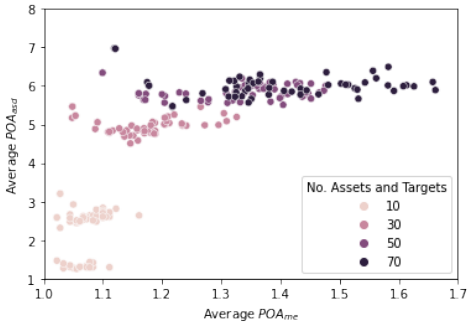
Figure 5. The effect of λ on HATRH-implemented Mosaic Warfare performance for Scenario 1, over 30 instances of each of 228 combinations of relative asset congestion, ϕ, and λ

vidual asset agency within Mosaic Warfare should be limited within high congestion instances akin to Scenario 1 if distance (fuel) efficiency is important to a decision-maker. In contrast, Figure 5b exhibits a coarse trend of decreasing POA_{me}-values with an increase in λ for a given level of relative asset congestion. This result implies that individual asset agencies should be encouraged within the same high congestion instances akin to Scenario 1 if the efficient use of munitions is of paramount importance. Thus, the λ-value should be determined based on the decision-maker's relative priority over the performance metrics. In the absence of a known priority, these results indicate $\lambda \in [0.6, 0.8]$ portend better outcomes.

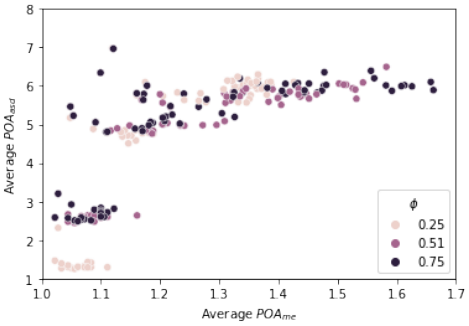
2.3.2 Testing Results for Scenario 2

Figure 6 depicts the testing results for Scenario 2, with Subfigures 6a-6c respectively differentiating the instances by the relative asset congestion, tile movement parameter, and asset movement parameter. For POA_{asd} calculations, Gurobi identified an optimal solution to Problem P for every instance, indicating assuredly accurate POA_{asd} values.

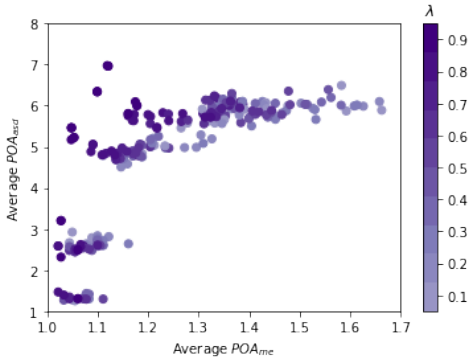
Selected performance trends from Scenario 1 remain present in the Scenario 2 results. Within Figure 6a, an increase in relative asset congestion tends to increase average POA_{asd}-values, at least initially. There is no exhibited difference in aver-



(a) Differentiated by relative asset congestion



(b) Differentiated by tile movement parameter (ϕ)



(c) Differentiated by asset movement parameter (λ)

Figure 6. HATRH-implemented Mosaic Warfare performance for Scenario 2, over 30 instances of each of 228 combinations of relative asset congestion, ϕ, and λ

age POA_{asd}-values between the 50 and 70 levels of relative asset congestion. This scenario-specific difference implies there may be some upper bound on the routing-related level of inefficiency for a defensive encirclement scenario as asset congestion increases. The effect of increasing relative asset congestion on average POA_{me} is again mild, slightly increasing the minimum, median, and maximum values.

Figure 6b exhibits very slight relationships between ϕ and the performance metrics for Scenario 2 instances. No relationship is evident for average POA_{asd}. However, an increase in ϕ exhibits a greater minimum, median, and maximum average POA_{me} for some but not all relative asset congestion levels. As with Scenario 1, there is a dubious benefit to frequent regrouping in the absence of special circumstances such

as losing assets due to adversary action.

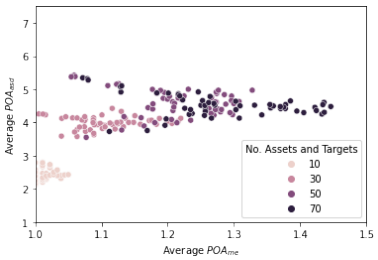
Figure 6c depicts an influence of λ on Mosaic Warfare performance consistent with the results observed for Scenario 1. For a given relative asset congestion level, higher λ-values portend lesser munition expenditures (i.e., lower POA_{me}-values). An increase in λ initially decreases and then gradually increases the average POA_{asd} for a given relative asset congestion level. Thus, the tradeoff in metric performance remains evident for λ. Although not presented here for the sake of brevity, extended testing on λ-values for Scenario 2 again demonstrated a range of $\lambda \in [0.6, 0.8]$ to be preferable, in the absence of a strict preference over the metrics.

2.3.3 Testing Results for Scenario 3

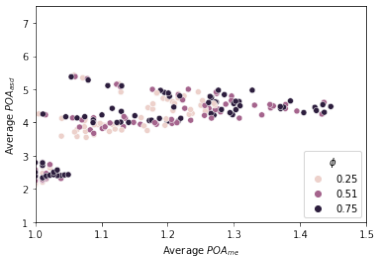
Figure 7 presents the testing results for Scenario 3, an offensive encirclement. Figures 7a-7c respectively differentiate the instances by the relative asset congestion, tile movement parameter, and asset movement parameter. For POA_{asd} calculations, Gurobi identified an optimal solution to Problem P for all but the instances having 70 assets-and-targets, for which it found feasible solutions with an average relative optimality gap of 3.3%. As such, the POA_{asd} metrics for those instances are (relatively tight) lower bounds on the values depicted in Figure 7.

Within Figure 7a, an increase in the relative asset congestion degrades both Mosaic Warfare performance in a manner consistent with Scenarios 1 and 2, as measured by POA_{asd} and POA_{me}. Worse performance results from a greater number of tiles making myopic decisions.

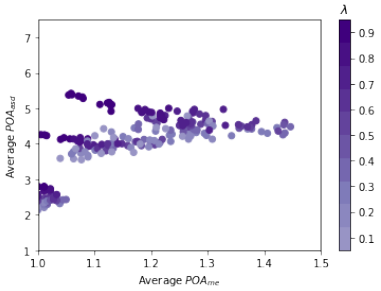
Relative to Scenarios 1 and 2, the effect of ϕ for a fixed, relative asset congestion level is slight and arguably negligible within Figure 7b, reinforcing the intuition that a regrouping procedure may not be warranted when all assets remain intact. Likewise, increases λ beyond a threshold can increase POA_{asd} while decreasing POA_{me}.



(a) Differentiated by relative asset congestion



(b) Differentiated by tile movement parameter (ϕ)

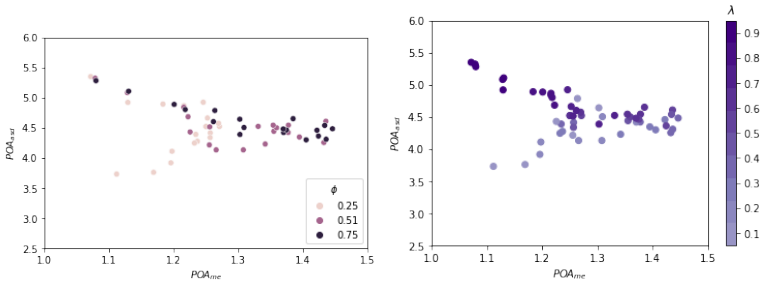


(c) Differentiated by asset movement parameter (λ)

Figure 7. HATRH-implemented Mosaic Warfare performance for Scenario 3, over 30 instances of each of 228 combinations of relative asset congestion, ϕ, and λ

However, these trends are not as strong for Scenario 3, and testing results exhibit some nuances worth additional exploration and discussion.

Figures 8a and 8b depict the results from Figures 7b and 7c, but with the results reduced to only the instances having a total of 70 assets and targets.



(a) Differentiated by tile move-ment parameter (ϕ) in relative high congestion

(b) Differentiated by asset move-ment parameter (λ) in relative high congestion

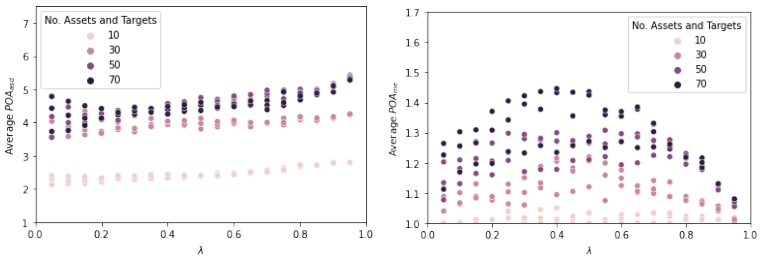
Figure 8. HATRH-implemented Mosaic Warfare performance for Scenario 3, in relatively high congestion (level 70), of ϕ, and λ

Note within Figure 8a that instances having lower average POA_{me}-values do not always exhibit an increase in average POA_{asd}. For instances with $\phi = 0.25$ and $POA_{me} \leq 1.2$, there are two subsets of outcomes, one each with higher and lower POA_{asd} values. Examining Figure 8b, it is evident that an interaction of λ and ϕ affects Mosaic Warfare performance. More specifically, low λ-values paired with low ϕ-values correspond to the lower POA_{asd}-values, whereas higher λ-values paired with the same low ϕ-values yield the higher of POA_{asd}-values.

This result indicates that the selection of the asset movement parameter λ should be considered in combination with selecting the tile movement parameter ϕ for high relative asset congestion levels. More notably, for Scenario 3, the previous testing-informed intuitions about the relative unimportance of regrouping assets and the proposed default value of $\lambda \in [0.6, 0.8]$ no longer hold. When relative asset congestion levels are high and the HATRH does not regroup assets frequently, less agency should be afforded to individual asset movement, else greater levels of inefficient asset routing will result.

As with Scenario 1, testing indicated merit to a more detailed examination of the effect of λ on solution performance for Scenario 3 instances. Figures 9a and 9b respectively depict the average POA_{asd}- and average POA_{me}-values for increasing values of $\lambda \in \{0.05, 0.1, ...0.95\}$ and, at each λ-value, for 30 instances each of the nine combinations of ϕ and four relative asset congestion settings. Both figures differentiate the instance performances by the relative asset congestion.

Within Figure 9a, the effect of λ on POA_{asd} is not as great for Scenario 3 instances. Moreover, the observed maximum POA_{me}-values decrease with either lower or higher λ-values, relative to $\lambda \approx 0.4$. In more detail, we observe the departure from previous intuition formed for Scenarios 1 and 2. When relative asset congestion is high, the asset movement parameter λ should be selected with deliberateness. If ϕ is lower,



(a) POA_{asd} **(b)** POA_{me}

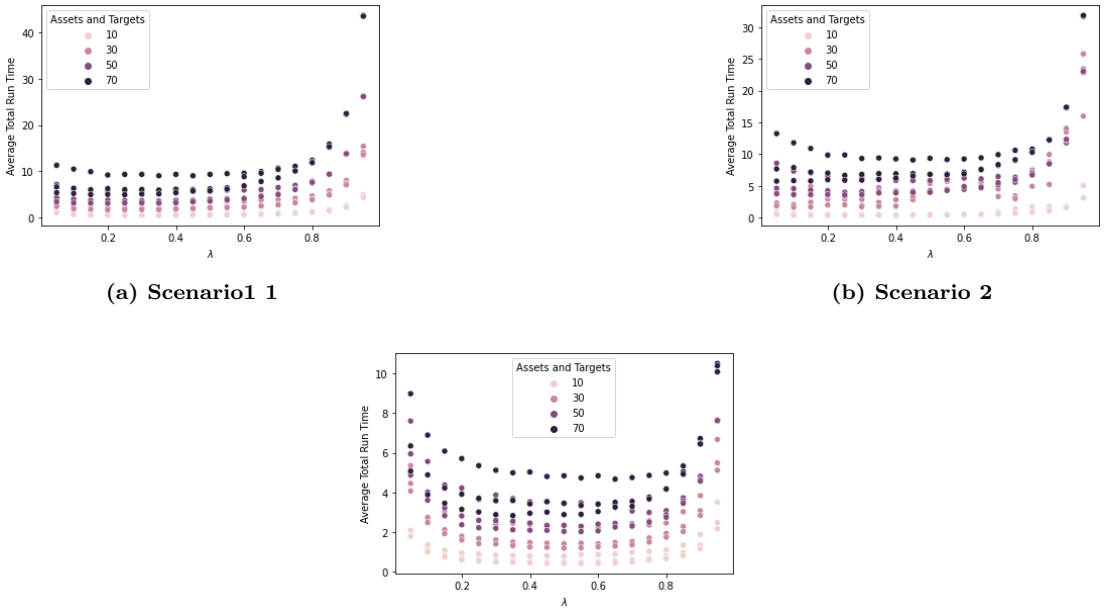
Figure 9. Asset movement parameter (λ) versus average POA_{asd} and POA_{me} differentiated by relative asset congestion

we recommend higher λ-values, but lower λ-values may yield superlative aggregate performance otherwise.

2.3.4 HATRH Computational Run time

The computational effort required by the HATRH was small (i.e., less than 0.6 seconds per iteration for the largest instances), indicating its suitability to rapidly assess the effect of different algorithmic parameters and problem features on the relative efficacy of Mosaic Warfare, relative to centralized command-and-control of assets when inter-asset communication is possible between all friendly assets. Across all scenarios, the computational time of HATRH increased with an increase in the relative asset congestion and an increase in λ, whereas ϕ exhibited no discernible effect. Figure 10 depicts the average computational time for each of the scenarios, as affected by both λ and the number of assets and targets.

Across all scenarios, a more congested operational space yielded higher relative average run time. Initially, an increase the value of λ decreases of average HATRH computational time. Whereas this decresase is more pronounced in Scenario 3, as depicted in Figure 10c, the average computational time is largest for Scenario 1 for comparative values of λ. Average HATRH run time reaches a minimum when $\lambda \approx 0.5$, then begins to increase as the values approaches one. Figure 10b shows the rapid



(a) Scenario1 1

(b) Scenario 2

(c) Scenario 3

Figure 10. Average Computational Time of HATRH by Scenario Relative to λ

increase in time in Scenario 2 for these values. In general, this result comports with expectations. Lower λ-values accord with vacillation; individual assets exhibit very little initiative, so the completion of a mission is left to the initiative of the command-and-control assets within the groups. In contrast, higher λ-values accord with aggressive chaos; larger initiative by individual assets can create tension with direction given by C2 assets.

The average time (in seconds) per iteration of HATRH increases roughly linearly with an increase in the number of assets and targets. Such a result also comports with expectations. The time required for an HATRH iteration depends directly on the number of assets that must be proximally (re)grouped. Finally, over the 6840 instances tested for each of Scenarios 1-3, the average of all run times were 5.6, 5.8, and 2.9 seconds, and the maximum run times were 43.7, 31.8, and 10.5 seconds. In general, these computational times are not cumbersome for HATRH to represent the implementation of Mosaic Warfare for these scenarios and instance sizes.

Of interest is the identification of when instance sizes may induce a notable computational burden. Examining the slowest performing instance of the HATRH for Scenario 1 with $\lambda = 0.95$, and $\phi = 0.51$, Table 1 presents the average HATRH run time and iteration count of 10 instances for all combinations of 100, 300, and 500 assets and targets with an operational area length of 100, 300, and 500.

Table 1. Computational Exploration of the HATRH

Total Assets and Targets	Total Operational Field Length	Average Total HATRH Run Time	Average Total HATRH Iterations
100		38.1	33.1
300	100	399.9	28.5
500		1573.5	28.8
100		214.8	178.9
300	300	2056.2	158.9
500		5995.5	151.1
100		440.1	398.2
300	500	4140.5	315.6
500		12669.0	306.2

For this greater number of assets and targets, and for the algorithmic parameters yielding the slowest performance in previous testing, both the average total time (in seconds) and the average time per iteration (in seconds) of HATRH increase at a slightly faster than linear rate as the number of assets and targets increases. The former trend provides greater context to the previous observation, as the linear trend only held for a smaller number of assets and targets. An increased operational area length will require more HATRH iterations to complete the mission because the aircraft movement within each iteration is limited. Overall, large instances involving hundreds of assets and targets operating on a large area requires more time to run HATRH without utilizing parallelization of selected processes within the overall algorithm.

2.4 Conclusions and Recommendations

This research evaluated a practical implementation of Mosaic Warfare, an operational concept conceived to implement decentralized, regional command-and-control over different types of warfighting assets in circumstances when centralized control is not feasible. To accomplish this, this research set forth a Hierarchical Asset Tiling and Routing Heuristic (HATRH) to emulate the principles of Mosaic Warfare by iteratively forming functional tiles of different types of assets, moving the tiles towards their myopically-identified targets, and allowing some user-defined degree of individual asset agency for movement, relative to its tile. Unique to this problem framework is the requirement to ensure tiles are functional in terms of composition and inter-asset communication; for the problem of interest, each tile must have a sensor, strike asset, and command-and-control asset to respectively identify a target, direct action, and implement that action. Informing both the testing herein and future evaluations of Mosaic Warfare, this research also proposed two performance metrics related to the efficient routing of assets and the efficient use of munitions to evaluate implementation vis-á-vis an approach leveraging centralized command and control.

Testing examined combinations of selected HRCH algorithmic parameters and problem features related to the engagement scenario and the relative asset congestion within an engagement area. Over all scenarios explored, the HATRH-implemented Mosaic Warfare was less efficient as relative asset congestion increased. A user-defined parameter corresponding to the frequency with which asset grouping decisions were revisited presented little-to-no effect on performance. However, a user-defined parameter that preserves some agency for individual assets to move relative to their tile did affect performance. Testing identified recommended a default range of this HATRH parameter setting, specific to each operational scenario.

The HATRH works well as a means to demonstrate and evaluate Mosaic War-

fare. Although problem instance sizes occasionally challenged the computation of performance metrics for higher relative asset congestion, it never degraded the performance of HATRH. Moreover, it is important to recall that HATRH accurately implements and executes in a central manner decisions that are, in practice, made in a distributed manner. Thus, the insights regarding performance pertain to Mosaic Warfare, whereas any computational challenges pertain only to this mechanism to synthetically emulate and evaluate Mosaic Warfare.

Two immediate directions for future research are readily apparent. It is worth testing the HATRH implementation of Mosaic Warfare in circumstances that may challenge existing insights. Specifically, extending testing may consider the loss of friendly assets, as it may demonstrate circumstances when more frequent regrouping decisions are beneficial. Moreover, variable algorithmic parameters should be considered; the effectiveness of Mosaic Warfare may improve if the relative aggressiveness and autonomy afforded to tile and individual asset movement adjust according to the proximity of tiles to targets-of-interest.

Alternatively, the analytically-driven insights compel the examination of different modeling approaches altogether. Because more frequent regrouping did not exhibit notable benefit, it may not be necessary. Moreover, a functional tile of assets need not form until it is time to identify and strike a target. Thus, an optimal routing and covering model is worth examining, with side constraints to enforce the proximal relationships necessary for different types of assets to be locally functional and with demand coverage (i.e., target destruction) occurring, e.g., within predesignated time windows. Computational challenges will accompany such an approach, and mitigation measures will be necessary.

III. A Multiple Asset-type, Collaborative Vehicle Routing Problem with Proximal Servicing of Demands

3.1 Introduction

This research explores the problem of routing assets having different capabilities over a network to service a set of demands in a collaborative manner, wherein the collaboration by different asset types entails nearly simultaneous service to a demand and a subset of asset types need only be close to a demand to service it.

Consider the following motivating civil scenario. During a heavy storm, weather radar detects several tornadoes that touch down within a region, temporarily disrupting cell phone communications, damaging buildings, and likely inducing human casualties. The threat of more tornadoes abates, and it is time to provide help with medical teams, humanitarian assets to deliver emergency provisions (e.g., food, water), and construction crews to begin rebuilding damaged infrastructure. These teams originate from different locations and can only communicate at limited ranges with shortwave radio. While it is possible to route the different asset types independently over the road network to visit the locations and provide service, a coordinated response is necessary because the services require collaboration. Moreover, when demand for relief and disaster supplies reach critical levels, rapid relief via collective provision is more important than any consideration of precedence. However, the different asset types should visit each location in a nearly simultaneous manner. A first advantage of this nearly simultaneous service is the completion of services in a coordinated manner so relief efforts can subsequently focus on other demands, elsewhere. In the absence of cell phone communications for coordination, police can provide local communication and direction of different assets supporting multiple demands. Moreover, police need only be within radio range of the demand locations during servicing

to direct the other asset types. A second advantage of nearly simultaneous service is that assets providing proximal service such as police are nearby to provide further, on-call assistance (e.g., security, law enforcement) to other asset types if a situation requires it (e.g., civil unrest).

This realistic scenario occurs with sufficient frequency to validate the importance of this research that seeks to identify a model and accompanying solution methodology to rapidly identify high quality solutions to route assets of different types and service demands in a collaborative manner, given the problem's complicating characteristics.

3.1.1 Literature Review

There is no shortage of research for routing disaster relief (e.g., Luis et al., 2012; Baxter et al., 2019; Liu et al., 2020; Chang et al., 2022). However, given the many complicating aspects of the asset routing problem this research seeks to address, a thorough review of the literature understandably did not identify research that addressed our problem.

Our problem is a variant of a route relaxed vehicle routing problem (VRP) (Goel and Gruhn, 2008; Goel, 2010), allowing assets to start and end at different locations. Toth and Vigo (2002) and Toth and Vigo (2014) published a sequence of diverse studies that collectively address common classes of VRPs and related solution methods. Golden et al. (2008) and Golden et al. (2023) produced another high-quality sequence of collected works in VRP research. Of interest is VRP literature related to the defining aspects of our problem: different types of assets that collaborate, (adjustable) time windows for servicing demands, and proximal service of demands.

Although a notable body of VRP research routes different types of assets, the manner in which they work together differs from our framework. Two-echelon VRPs (Li et al., 2021; Sluijk et al., 2022) coordinate different asset types, e.g., one type

transports goods from warehouses to satellite locations, and another type transports goods to customers. Split-delivery VRPs (Ceselli et al., 2009; Santillán et al., 2012) are also related. Although this variant does not route different types of assets, it does consider different assets working together to service demands via the accumulation of partial servicing of demands by each of multiple, limited capacity assets. Missing from these categories of VRPs are multiple assets types wherein each asset type must service a demand for that demand to be serviced.

The VRP with Time Windows (VRPTW) provides a modeling structure with which we can enforce the requirement that different types of assets must service a demand with near simultaneity. The VRPTW routes assets to service demands within their respective, predetermined time windows (e.g., Desrochers et al., 1992; Bräysy and Gendreau, 2005*a*,*b*). Within VRPTW research, modeling structures vary according to the nature of the time windows (Toth and Vigo, 2014). Time windows may be narrow or wide, and time window constraints may be softly enforced (Cordeau et al., 2002). Strictly enforced time windows can represent nearly simultaneous servicing of demands by different asset types, but the *a priori* identification of time windows is problematic, given the need to coordinate the routing of different numbers of assets of varying types traveling at different speeds over a network from their respective origins. Departing from the traditional use of time windows within the VRP literature, our model uses them only to enforce nearly simultaneous service, not any servicing deadlines for demands. Thus, it is permissible for the time windows to shift. Given this context, modeling in Section 3.2 considers adjustable time windows, a feature we have not identified in the literature. As an aside, several VRPTW studies route different asset types (e.g. Chen et al., 2021; Kuo et al., 2022), but such research typically examines two-echelon routing problems, again belying the nature of collaborative asset type interactions in our underlying problem.

There is some VRP literature that considers proximally-serviced demand. Recent work by Frey et al. (2022) explored a VRPTW variant that allows for flexible delivery locations (VRPTW-FL) for each demand. In a compelling work, Tilk et al. (2021) investigated the vehicle routing problem with delivery options (VRPDO), wherein different delivery location and time window options exist for each demand. Both of these works considered a single type of vehicle delivering a common type of asset, so they were absent any collaborative service *specific to each demand*. Modeling in Section 3.2 does not leverage the latter authors' representation of alternative time windows because it would increase the formulation size too much, but their mutual embrace of alternative locations to service demand conceptually informs the representation of proximal service for modeling in Section 3.2, at least indirectly.

Within the context of the related VRP literature, the defining aspects of our problem are unique. The manner of collaboration between different asset types is novel. Although VRPTW structures can help enforce the collaboration between asset types when servicing demands, the adjustable nature of time windows will be new. Likewise, although the VRPDO structure for alternative service times is not suitable, both it and the VRPTW-FL inspire an approach to represent proximal servicing of demands.

3.1.2 Statement of Contributions

This research makes two contributions to the VRP literature. First, it sets forth a mixed-integer linear programming model to route different asset types that must collaboratively service demands with near simultaneity, as represented with adjustable time windows, and while allowing a subset of assets to service demands proximally rather than via co-location. Second, it proposes customized solution method alternatives comprised of combinations of permutations of a model decomposition heuristic

and either of two preprocessing techniques to improve the likelihood of the heuristic identifying a feasible solution.

Hereafter, Section 3.2 introduces the proposed model with its underlying assumptions, and Section 3.2.2 presents the model decomposition heuristic and the preprocessing techniques. Section 3.4 illustrates the model validity via a representative instance, after which it reports the result of computational testing to compare the nine combinations of solution methods and preprocessing techniques that collectively demonstrates their relative efficacy and tractability. Finally, Section 3.5 concludes with a summary of modeling and computational insights, and proposes directions to extend this research.

3.2 Model Formulation

Section 3.2.1 discusses modeling assumptions, and Section 3.2.2 presents the mixed-integer linear programming formulation for the problem of interest. The modeled vehicle routing problem variant services each demand by an asset from each of multiple asset types, and it allows a subset of asset types to service demands proximally (i.e., within a given distance of the demand rather than by visiting it). To enforce collaborative servicing, the model uses time windows, wherein collaborative servicing occurs because an asset of each type services a demand a within a time window. Additionally, the model allows for the adjustment (i.e., shifting) of each demand-specific time window to coordinate the timing with which demands are serviced by assets of different types.

3.2.1 Modeling Assumptions

A few assumptions are foundational to our modeling. First, we assume the assets are routed on a network comprised of nodes and arcs. This assumption is directly

representative when modeling a related problem for a road network, and it likewise holds when allowing assets to traverse directly between demand points in Euclidean space. Alternatively, for problems allowing direct, point-to-point travel, it enables the discretization of Euclidean space via tessellation. The ability to discretize the space in this manner enables proximal service; an asset with such a capability can service a demand from a proximal node in the network. Second, we assume that different asset types may service a demand with near simultaneity rather than either exact simultaneity or in accordance with a strict precedence of their on-site role in service. We contend this assumption is acceptable because we do not expect the prescribed solution for routing vehicles to be implemented in exact measure; in practice, there is variability in travel time, time on-site, and the like. Third, the duration of time to service a demand is negligible. For some applications (e.g., delivery of construction supplies), this assumption is representative; for other applications (e.g., disaster relief), it is a notably simplifying assumption. The current model can be parameterized to negate the simplification implied by this assumption, as will be discussed in Section 3.2.

3.2.2 Mathematical Program

As an introduction to the model formulation, it is first important to define the relevant sets, parameters, and decision variables.

Sets

- Ψ: the set of asset types, indexed on ψ, where a demand must be serviced by each asset type in a nearly simultaneous manner
  - $\Psi^D \subseteq \Psi$: the subset of asset types that must visit a demand node to service it
  - $\Psi^P \subseteq \Psi$: the subset of asset types that can service demands proximally. Assumed is $\Psi^D, \Psi^P \neq \emptyset$, else a simpler model is appropriate.

- K: the set of assets, indexed on k and partitioned on Ψ, with subsets $K_\psi \neq \emptyset,\ \forall\ \psi \in \Psi$
- N: the set of nodes in the network, indexed on either i or j
  - $N^d \subseteq N$: the subset of nodes that are demand nodes
  - $N^d_{i\psi} \subset N$: the subset of nodes from which asset type $\psi \in \Psi^P$ can proximally service demand node $i \in N^d$
- A: the set of directed network arcs, indexed on (i, j)
- $G(N, A)$: the directed network comprised of nodes N and arcs A

Parameters

- c_{ijk}: a per unit cost incurred by asset $k \in K$ traversing arc $(i, j) \in A$
- b_{ik}: a binary parameter equal to 1 if asset k is initially located at node i, and 0 otherwise
- τ_{ijk}: a positive parameter indicating the minimum time required to transverse arc (i, j) for asset k
- $\overline{LB_i}$: a lower bound representing the earliest time at which an asset of any type $\psi \in \Psi$ may service a demand at node $i \in N^d$
- $\overline{UB_i}$: an upper bound representing the latest time at which demand $i \in N^d$ must be serviced by all asset types, if any asset type services it at time $\overline{LB_i}$. The computation $\left(\overline{UB_i} - \overline{LB_i}\right)$ is the *service window width* of time representing near simultaneity.
- ω: a scalar representing how much slower an agent may traverse an arc. For example, if $\omega = 2$, an agent may take up to twice as long as their fastest time to traverse an arc, i.e., their slowest speed is $1/\omega$ times their fastest speed.
- δ: the increment of time with which the lower and upper bounds on a demand-specific time window may be increased, i.e., the *service window shift increment*. Of note, by setting the $\overline{LB_i}$-values for an instance to be the soonest any asset can service demand i, there is neither need nor merit to decrementing the bounds for a service time window.
- M_1, M_2: large, positive numbers used within bounding constraints in the formulation. Herein, it suffices to set $M_1 = |N|$ and $M_2 = \tau^{max}_{ijk} \cdot |A|$, where $\tau^{max}_{ijk} = \underset{(i,j)\in A, k\in K}{\text{argmax}} \{\tau_{ijk}\}$.

Decision Variables

- x_{ijk}: a binary decision variable equal to 1 if asset k traverses arc (i, j), and 0 otherwise
- t_{ik}: a non-negative decision variable indicating the time at which asset k arrives at node i
- α_i: a non-negative decision variable representing the number of shifts of duration δ applied to the time window for servicing node $i \in N^d$. For computational tractability, we assume $\alpha_i \in \mathbb{Z}_+,\ \forall\ i \in N^d$.
- ϕ_{ijk}: a binary decision variable equal to 1 if asset $k \in K_\psi$ for $\psi \in \Psi^P$ services demand node $i \in N^d$ from proximal node $j \in N^d_{i\psi}$, and 0 otherwise
- u_{ik}: a non-negative decision variable specific to each node i and asset k, used to implement Miller-Tucker-Zemlin (MTZ) subtour elimination constraints (Miller et al., 1960).
- LB_i: a non-negative decision variable equal to $\overline{LB}_i + \delta\alpha_i$, indicating the earliest time any asset type can service demand node $i \in N^d$
- UB_i: a non-negative decision variable equal to $\overline{UB}_i + \delta\alpha_i$, indicating the latest time every asset must service demand node $i \in N^d$

Given this notation, the math programming formulation for the collaborative vehicle routing problem with proximity service (**CoVRP-PS**) follows.

$$\min_{\mathbf{x},\mathbf{t},\alpha,\phi} \sum_{(i,j)\in A} \sum_{k\in K} c_{ijk} x_{ijk} \tag{11}$$

$$\text{s.t.} \sum_{j:(i,j)\in A} x_{ijk} - \sum_{j:(j,i)\in A} x_{jik} \leq b_{ik},\ \forall\ i \in N, k \in K, \tag{12}$$

$$\sum_{j:(j,i)\in A} \sum_{k\in K_\psi} x_{jik} \geq 1,\ \forall\ i \in N^d, \psi \in \Psi^D, \tag{13}$$

$$\sum_{\substack{j\in N,\\ i'\in N^d_{i\psi}:(j,i')\in A}} \sum_{k\in K_\psi} x_{ji'k} \geq 1,\ \forall\ i \in N^d, \psi \in \Psi^P, \tag{14}$$

$$u_{ik} - u_{jk} + M_1 x_{ijk} \leq M_1 - 1,\ \forall\ (i,j) \in A, k \in K, \tag{15}$$

$$t_{ik} + \tau_{ijk} \leq t_{jk} + M_2(1 - x_{ijk}),\ \forall\ (i,j) \in A, k \in K, \tag{16}$$

$$t_{jk} \leq t_{ik} + \omega\tau_{ijk} + M_2(1 - x_{ijk}),\ \forall\ (i,j) \in A, k \in K, \tag{17}$$

$$t_{ik} \leq M_2 \sum_{j:(j,i)\in A} x_{jik},\ \forall\ i \in N, k \in K, \tag{18}$$

$$t_{ik} \leq UB_i,\ \forall\ i \in N^d, k \in K_\psi, \psi \in \Psi^D, \tag{19}$$

$$LB_i \leq t_{ik},\ \forall\ i \in N^d, k \in K_\psi, \psi \in \Psi^D, \tag{20}$$

$$\sum_{j \in N^d_{i\psi}} \sum_{k \in K_\psi} \phi_{ijk} = 1,\ \forall\ i \in N^d, \psi \in \Psi^P, \tag{21}$$

$$LB_i - M_2(1 - \phi_{ijk}) \leq t_{jk}\ \forall\ i \in N^d, j \in N^d_{i\psi}, k \in K_\psi, \psi \in \Psi^P, \tag{22}$$

$$t_{jk} \leq UB_i + M_2(1 - \phi_{ijk})\ \forall\ i \in N^d, j \in N^d_{i\psi}, k \in K_\psi, \psi \in \Psi^P, \tag{23}$$

$$LB_i = \overline{LB_i} + \delta\alpha_i,\ \forall\ i \in N^d, \tag{24}$$

$$UB_i = \overline{UB_i} + \delta\alpha_i,\ \forall\ i \in N^d, \tag{25}$$

$$x_{ijk} \in \{0,1\},\ \forall\ (i,j) \in A, k \in K, \tag{26}$$

$$t_{ik} \geq 0,\ \forall\ i \in N, k \in K, \tag{27}$$

$$\alpha_i \in \mathbb{Z}_+,\ \forall\ i \in N^d, \tag{28}$$

$$\phi_{ijk} \in \{0,1\},\ \forall\ i \in N^d, j \in N^d_{i\psi}, k \in K_\psi, \psi \in \Psi^P, \tag{29}$$

$$u_{ik} \geq 0,\ \forall\ i \in N, k \in K. \tag{30}$$

The formulation seeks to minimize the total travel costs of all assets traversing the network (11). Constraint (12) enforces the conservation of flow for the movement of all assets without requiring assets to return to their respective origins. Constraint (13) ensures that each demand node is visited by at least one asset k of each type $\psi \in \Psi^D$ that cannot provide service proximally. Constraint (14) ensures that at least one asset k of each type $\psi \in \Psi^P$ visits a node $i' \in N^d_{i\psi}$ from which it can proximally service demand $i \in N^d$. Constraint (15) applies the MTZ subtour elimination constraints.

Constraints (16) and (17) calculate the time of each asset as it moves through the network. Within Constraint (17), the term $\omega\tau_{ijk}$ documents an implied assumption

that assets may slow down to $1/\omega$ of the fasted speed at which they can traverse arc (i, j) to coordinate synchronize times at demands. An alternative assumption is possible and should be application-specific. Constraint (18) ensures any non-moving asset does not receive a positive t_{ik}-value for any node, a constraint imposed to prevent solvers from reporting alternative basic feasible solutions that do not represent time accurately.

Constraints (19) and (20) collectively impose the time window for every asset type $\psi \in \Psi^D$ to visit demand node $i \in N^d$. For asset types $\psi \in \Psi^P$, Constraint (21) identifies the proximal node $j \in N^d_{i\psi}$ from which asset $k \in K_\psi$ provides service to demand node $i \in N^d$. Such a decision imposes the time window for proximal service via Constraints (22) and (23).

Finally, Constraints (24) and (25) compute the lower and upper bounds for adjustable time windows for each demand node, and Constraints (26)-(30) enforce the respective, appropriate restrictions on the decision variables.

The resulting CoVRP-PS formulation is directly solvable via any of a number of readily available, commercial solvers. However, the NP-hard nature of the VRP, for which the nuances of this formulation are expected to further complicate, motivates the development of customized solution methods to improve computational tractability over what a commercial solver will attain.

3.3 Solution Methodology

Preliminary testing of CoVRP-PS instances using the solver Gurobi (Version 9.5.1) confirmed the aforementioned intuition regarding computational challenges, compelling the design and evaluation of alternative solution methodologies.

Figure 11 depicts several alternative solution methodologies to solve CoVRP-PS instances. An answer of 'no' to every question entails solving an instance directly, as

formulated. An answer of 'no' to the first question and 'yes' to a decomposition of the problem by asset type entails solving the problem in two stages, for which Section 3.3.1 details two alternatives. These heuristics can be enhanced with a preprocessing technique to impose lower bounds on the α_i-values, and Section 3.3.2 sets forth two such techniques. Because complications arising from the model decomposition approach motivates the use of preprocessing techniques, Section 3.3.1 discusses the former heuristic component before Section 3.3.2 details the latter component.

Additionally, Section 3.3.3 identifies a fundamentally different heuristic approach entailing a maximal decomposition of the problem. We detail the conceptual procedure herein, but we set it aside for testing as a sequel effort to this dissertation.

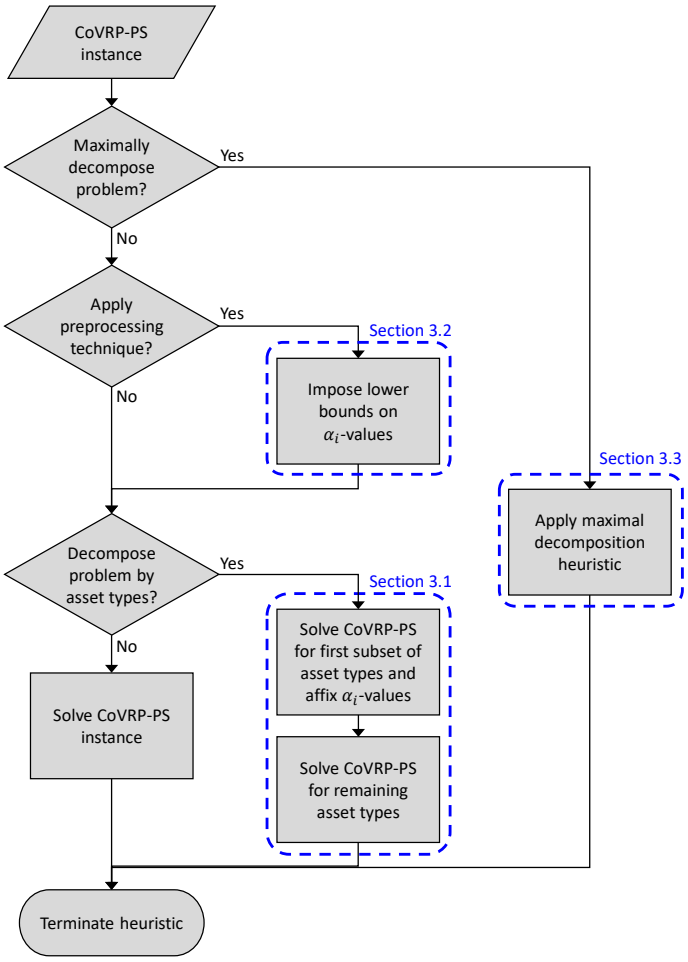


Figure 11. Solution methodology flowchart

3.3.1 Two-stage Model Decomposition Heuristic

Notable complicating aspects of the CoVRP-PS model are the different types of assets Ψ^D and Ψ^P, and the different manners in which those asset types can service demands. Proximal service capabilities require a non-trivial number of additional ϕ_{ijk}-variables. Logically, solving the problem for only the asset types Ψ^D should be much easier.

Accordingly, testing in Section 3.4 considers a model decomposition heuristic that first routes the asset types in Ψ^D, solving CoVRP-PS in the absence of Ψ^P-specific decision variables and constraints. Affixing the α_i variables from the solution as parameters, it subsequently routes the asset types Ψ^P in the absence of decision variables and constraints related to adjusting the time windows. An alternative permutation sequentially routes asset types Ψ^P, affixes the α_i-variables, and routes asset types Ψ^D. This latter approach is not expected to perform as well, given the first problem must address the complications of both adjustable time windows and proximal service in the first problem. In contrast, routing asset types Ψ^D before asset types Ψ^P allocates these two computationally challenging model characteristics to two different math programs.

By decomposing the problem, this solution method entails a heuristic approach to solve a CoVRP-PS instance because an optimal solution might not be attainable. The myopic, optimal routing of the first subset of asset types may affix α_i-values suboptimally with respect to the original problem.

More challenging, initial testing of this heuristic identified its potential failure to identify a feasible solution. Given a feasible CoVRP-PS instance, the problem of routing the first subset of assets is always feasible. However, the affixed α_i-values may render the subsequent routing problem(s) infeasible. Instance characteristics that made this outcome more likely were either routing proximally servicing asset

types first or having a fewer number assets of a given type – relative to the first asset type(s) considered – to route after α_i-values are affixed. In the former case, asset types providing proximal service may not need as much time to service all demands compared to asset types Ψ^D, and the resulting, affixed time windows may induce an infeasible, second-stage problem. In the latter case, without regard to the permutation of the model decomposition heuristic, affixed time windows determined by routing many assets in the first-stage problem may be infeasible for a lesser number of assets to be routed in the second-stage problem.

3.3.2 Preprocessing Techniques to Bound Service Time Window Shifts

Without loss of generality, denote the first and second sets of sequentially routed asset types via the model decomposition heuristic as Ψ^1 and Ψ^2. Initial testing identified the potential for a model decomposition approach to fail. That is, it sometimes identified an optimal CoVRP-PS solution for asset types Ψ^1, affixing demand-specific time windows such that the CoVRP-PS instance for routing asset types Ψ^2 was infeasible. This potential inability of the decomposition heuristic to identify a feasible solution motivates preprocessing techniques to impose lower bounds on the α_i-values.

We consider preprocessing techniques to impose lower bounds on α_i-values via Constraint 31 when routing asset types Ψ^1 such that, upon affixing the α_i-values, it is less likely that the problem of routing asset types Ψ^2 is infeasible. Any such preprocessing technique should be conceptually sound and computationally efficient.

$$\alpha_i \geq \alpha_i^{LB}, \ \forall \ i \in N^d \tag{31}$$

The objective of a preprocessing technique is to estimate the latest time by which at least one asset $k \in K_\psi$ of every type $\psi \in \Psi^2$ can service a demand at node $i \in N^d$. Define α_{ik} to be the least number of service time window shifts necessary for asset

$k \in K_\psi, \psi \in \Psi^2$ to service demand $i \in N^d$; and $\alpha_{i\psi} = \min_{k \in K_\psi}\{\alpha_{ik}\}$ to be the least number of service time window shifts necessary for asset type ψ to service demand $i \in N^d$. By definition, Equation (32) identifies a valid α_i^{LB} that will not eliminate an optimal solution to the original CoVRP-PS instance.

$$\alpha_i^{LB} = \max_{\psi \in \Psi^2}\{\alpha_{i\psi}\} \tag{32}$$

Of note, Equation 32 may not generate tighter lower bounds on the optimal α_i-values for the original CoVRP-PS instance. As such, it is of interest to explore preprocessing techniques that overestimate α_i^{LB} to improve the likelihood of a decomposition heuristic identifying a feasible solution to the original problem, even though it may eliminate the optimal solution.

Remaining is to identify the means by which a preprocessing technique estimates the α_{ik}-values. Our research considers two conceptual approaches, hereafter identified by the routing technique used to compute α_{ik}-values: the Floyd-Warshall and Nearest Neighbor preprocessing techniques.

Floyd-Warshall Technique

This estimation technique assumes each asset $k \in K_\psi, \psi \in \Psi^2$ traverses the network at its slowest rate (i.e., $\tau_k = \max_{(i,j)\in A}\{\tau_{ijk}\}$) and travels directly to each demand node $i \in N^d$. It applies the Floyd-Warshall Algorithm (Floyd, 1962) to compute the minimum distance d_{ik} between every source node for an asset $k \in K_\psi, \psi \in \Psi^2$ and every demand node $i \in N^d$. Subsequently, it estimates α_{ik}-values via Equation (33).

$$\alpha_{ik} = \begin{cases} 0 & (d_{ik}/\tau_k) \leq \overline{UB} \\ \left\lceil \left((d_{ik}/\tau_k) - \overline{UB} \right) / \delta \right\rceil & \text{otherwise} \end{cases} \tag{33}$$

Admittedly, this technique is imperfect. First, it may overestimate α_{ik}-values because of its pessimistic assumption about the assets' common rate of travel. Second, it may *underestimate* α_{ik}-values by underestimating the distance an asset will travel to reach a demand $i \in N^d$ because the shortest paths between source nodes for assets and demand nodes do not account for the need to route assets to service multiple demand nodes.

For example, consider a singleton set K_ψ for an asset of type $\psi \in \Psi^2$. Given demand nodes $N^d = \{8, 9\}$ and asset $k = 7$, the Floyd-Warshall technique may calculate $d_{87} = 2$ and $d_{97} = 4$. However, asset $k = 7$ must service both demand nodes, and a shortest route would first visit Node 8 and subsequently travel 3 units of distance to reach Node 9. For such a case, the Floyd-Warshall technique would underestimate α_{97} by 25%. The optimal solution to the first-stage problem may affix the α_i-values too low, and the decomposition heuristic leveraged with this preprocessing technique may not be able to identify a feasible solution to the second-stage problem.

Alternatively, one could certainly calculate α_{ik}-values by separably routing every asset type $\psi \in \Psi^2$ via a much reduced version of CoVRP-PS, but we set aside this notion as counterproductive in terms of computational effort. Such an approach would involve solving $2 + |\Psi^2|$ math programs, none of which is a trivial endeavor.

Nearest Neighbor Technique

Seeking a fast technique to reduce the potential underestimate of α_{ik}-values, we leverage a Nearest Neighbor approach. For each $k \in K_\psi, \psi \in \Psi^2$, we apply the Nearest Neighbor heuristic (Fix and Hodges Jr, 1952) to visit every demand node. Defining d_{ik} as the distance travelled by asset k to reach node i, the procedure calculates the α_{ik}-values via Equation (33).

Enhancement of Both Preprocessing Techniques

Additionally imposed constraints in the first-stage problem better preserve time for assets types in the second-stage problem to service demand nodes. As a motivating example, consider $N^d = \{6, 7\}$ and $\alpha_6^{LB} = 4$ and $\alpha_7^{LB} = 2$ via the Equation 32. In the first stage problem, assets types Ψ^1 service the demands using time window shifts of $\alpha_6 = 4$ and $\alpha_7 = 4$. If those α_i-values are affixed for the second-stage problem, it may be infeasible because a narrow time window may not allow a fewer number of subsequently routed assets to service all demand nodes. Instead, it is relevant to preserve the *magnitude* of differences in α_i^{LB}-values as well.

We preserve this magnitude for both preprocessing techniques as follows. We first sort the demand nodes in ascending order of α_i^{LB}-values, as calculated via Equation (32). Denoting sequential pairs in this sorted list as $(i, i') \in S$, we define $\alpha_{ii'}^{offset} = \alpha_{i'}^{LB} - \alpha_i^{LB}$. Subsequently, we impose Constraint (34) in addition to Constraint (31) when solving the first-stage problem within the model decomposition heuristic.

$$\alpha_{i'} \geq \alpha_i + \alpha_{ii'}^{offset}, \ \forall \ (i, i') \in S \tag{34}$$

On the Use of Preprocessing Techniques for the General Problem

By leveraging a simple routing heuristic that may not find the minimal routing distances for asset k to service each demand, this preprocessing technique used in combination with a model decomposition heuristic may not identify an optimal solution to CoVRP-PS, but it will increase the likelihood of a feasible solution found in the second-stage problem.

Although the development of the preprocessing techniques is intended to help resolve identified issues with the model decomposition heuristic, they may also help a commercial solver identify a feasible CoVRP-PS instance more efficiently. Defining

$\Psi^1 = \Psi$ and $\Psi^2 = \emptyset$, either preprocessing technique can rapidly identify additional bounds on α_i-values for the CoVRP-PS formulation. Such bounds should reduce the feasible region and improve solver convergence, a conjecture that testing in the Section 3.4 examines.

3.3.3 Maximal Decomposition Heuristic

As an alternative to the heuristic described in Section 3.3.1, we also set forth a maximal decomposition heuristic that foremost emphasizes computational efficiency. It routes one asset type at-a-time, initially ignoring the requirement they collaborate. For each asset type, it first solves an assignment problem to assign demands to assets at their originating nodes while minimizing the average assignment distance. It subsequently routes each asset to service its assigned demands, either directly or proximally, as appropriate. Assembling the collective routing solutions for the different asset types, the heuristic determines whether the solution is feasible with respect to the adjustable time windows. As a broad characterization, we expect this heuristic to rapidly identify infeasible solutions more often than feasible solutions. As such, we forgo its testing in this dissertation, but we will explore it in a sequel effort to address a question raised by a reviewer from a journal for which this contribution is being considered for publication.

3.4 Testing, Results, and Analysis

Section 3.4.1 presents a CoVRP-PS instance and its optimal solution to illustrate the problem and motivate the subsequent testing regimen. Section 3.4.2 discusses the manner utilized to generating test instances and explores the effect of six problem factors on the required computational effort by a leading commercial software to solve CoVRP-PS instances directly, culminating with the presentation of a designed exper-

iment for comparative testing of alternative solution methods. For that experiment, Section 3.4.3 reports and discusses the detailed performance attained via the commercial solver, providing evidence of the need for alternative solution methodologies. Section 3.4.4 compares the performance of the nine alternative solution methods that leverage combinations of model decomposition heuristic permutations and preprocessing techniques. This section concludes with general insights regarding solution method use vis-á-vis CoVRP-PS instance parameter characteristics.

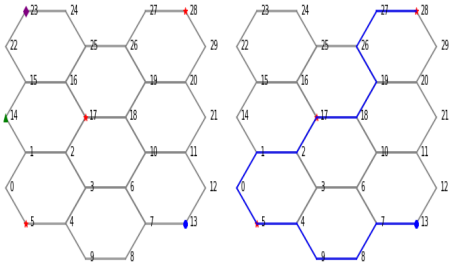
3.4.1 Illustrative CoVRP-PS Instance

We illustrate a representative instance of the CoVRP-PS formulation. This instance has $|\Psi| = 3$ asset types and $|K| = 3$ total assets to route and service $|N^d| = 3$ demands. For simplicity, $k = \psi,\ \forall\ k \in K$, i.e., $K_\psi,\ \forall\ \psi \in \Psi$ are singletons. As-set types $\Psi^D = \{1, 2\}$ must provide direct service to demands, whereas asset type $\Psi^P = \{3\}$ may service demands proximally by visiting an adjacent node in the network.

For a network $G(N, A)$, we consider a regular hexagonal tessellation of a planar region, a discretization technique embraced in the literature for routing assets not restricted to a road network (e.g., Lunday et al., 2012; Lessin et al., 2018), which is superior to alternative tessellations via squares or triangles for its balance between result model granularity and tractability (Yousefi and Donohue, 2004). Figure 12a depicts the network having $|N| = 30$ numbered nodes and $|A| = 36$ arcs. Within the figure, red stars denote the demand nodes $N^d = \{5, 17, 28\}$, whereas a purple diamond, blue circle, and green triangle respectively indicate the origins of assets $k = 1, 2, 3$ at Nodes 23, 13, and 14 (i.e., $b_{23,1} = b_{13,2} = b_{14,3} = 1$).

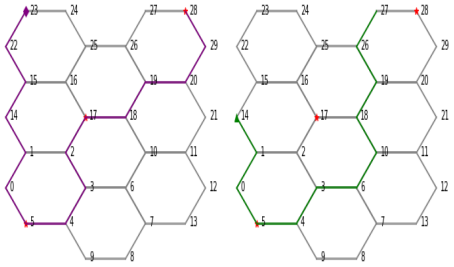
We affix $c_{ijk} = 1,\ \forall\ (i, j) \in A, k \in K$, making the objective function equivalent to minimizing the number of arcs traversed. Setting $\omega = 2$ allows assets to slow

down to half their fastest speed to trasverse an arc, as necessary. For asset arc traversal times, $(\tau_{ij1}, \tau_{ij2}, \tau_{ij3}) = (0.8, 1, 0.8)\,,\ \forall\ (i,j) \in A$, indicating asset $k = 2$ cannot travel as fast as the other assets. This instance has initial service windows of $\left[\overline{LB}_i, \overline{UB}_i\right] = [2, 4]\,,\ \forall\ i \in N^d$, and $\delta = 3$ as the service window shift increment. These latter parameters deliberately render certain time windows (e.g., $(4,5)$, $(7,8)$) infeasible for servicing any demand.



(a) Initial Position-ing

(b) $k = \psi = 1$ Routing



(c) $k = \psi = 2$ Routing

(d) $k = \psi = 3$ Routing

Figure 12. Illustrative Instance Network and Optimal Asset Routing

Figures 12b-12d respectively depict an optimal routing solution of assets 1, 2, and 3, and Table 2 reports the elements of the optimal solution related to the spatiotemporal servicing of demands. The first two rows of Table 2 present the number of time increment shifts applied for the respective demands, followed by the resulting service window. The next two rows report the time at which assets $k = 1, 2$ serviced the demands, and the final two rows indicate the node $j|\phi_{ij3} = 1,\ \forall\ i \in N^d$ from which asset $k = 3$ serviced the demands and the time at which it occurred.

Table 2. Demand-specific Characteristics for the Optimal Solution to the Illustrative Instance

Solution	**Demand Nodes N^d**		
Characteristic	$i = 5$	$i = 17$	$i = 28$
α_i	1	3	5
(LB_i, UB_i)	(5,7)	(11,13)	(17,19)
t_{i1}	7.0	12.6	17.0
t_{i2}	7.0	11	19.0
$j \vert \phi_{ij3} = 1$	4	18	27
t_{j3}	6.4	12.8	17.6

Within Figures 12b-12d, assets $k = 1, 2, 3$ respectively traversed 15, 14, and 11 arcs to service the demands, corresponding to $z^* = 40$. Whereas assets $k = 1$ and $k = 2$ visited each demand, asset $k = 3$ only directly visited the demand at node $i = 5$. However, asset $k = 3$ serviced every demand proximally, as the next-to-last row in Table 2 indicates. In fact, asset $k = 3$ travelled by demand node $i = 5$ before servicing it from node $j = 4$, although an alternative optimal solution of servicing it directly exists. Due to temporal constraints (19)-(23) requiring all assets arrive within a time window, asset $k = 3$ delays its servicing of demands and routes to the three demand nodes via a path that is not cost-minimizing. Compelling this outcome is the slower speed of asset $k = 2$, which arrives at the upper bound of the shifted time window determined for demand Nodes 5 and 28. Thus, slower assets may require faster assets to take longer routes to coordinate service times. Of note, only the width of the service time window for the demand at Node 28 was fully utilized, with assets arriving at the beginning ($k = 2$) and end ($k = 1$) of its shifted service window. This solution indicates the existence of alternative optima in the t_{ik}-space and t_{jk}-space, e.g., asset $k = 3$ did not need to service Node 28 at exactly $t_{27,3} = 17.6$.

Of related interest is the ability of the model to prescribe and modulate rates of travel for assets, as allowed via Constraints (16) and (17). Decomposing the paths depicted in Figures 12b-12d to the three demand-serving component paths, Table 3 reports the average speeds (i.e., arcs per time period) for each asset when traversing

the network to service each subsequent demand. Of note, no asset travelled at either its minimum or maximum speed when moving to service a demand, and only asset $k = 3$ travelled at a consistent average speed for each of the demands. This result confirms the existence of alternative optima, and extensions to the proposed model might consider additional objectives, such as minimizing the variation in speed for assets as they traverse the network.

Table 3. Number of Arcs Travel Per Time Period

	(**Min**,**Max**)	**Demand Serviced**		
Asset	**Arcs Traveled**	$i = 5$	$i = 17$	$i = 28$
1	(0.63, 1.25)	0.71	0.71	1.14
2	(0.50, 1.00)	0.86	1.00	0.63
3	(0.63, 1.25)	0.63	0.63	0.63

In general, the results in Figure 12 and Tables 2 and 3 collectively demonstrate the model's ability to prescribe an optimal solution to a CoVRP-PS instance that ensures collaborative service of demands by different asset types, a subset of which may service demands proximally.

3.4.2 Test Instance Generation and Computational Test Design

To assess the relative ability a commercial solver to solve CoVRP-PS instances directly and to compare its performance against alternative solution methods, it is important to identify the relevant problem features that affect solver performance. Exploratory testing identified six features suspected to affect solver convergence time. Subsequent testing examined the effect of the number of nodes in the network; the number of demand nodes; the number of assets of different types to route; the service window shift increment δ; the service window width $\left(\overline{UB}_i - \overline{LB}_i\right)$; and the earliest possible service time $\overline{LB}_i$. This section discusses that testing and the identified features to inform a designed experiment for comparative testing, as detailed in Section 3.4.4.

Testing utilized the following method to generate CoVRP-PS instances. Given a number of nodes $|N|$, the number of regular hexagons in a column of tessellated planar space is equal to the number of columns. In combination with this network design, $|N|$ induces the arcs A and network $G(N, A)$. (Of course, $|N|$-values under this scheme are limited to values that yield such a network.) The demand nodes N^d are located in the respective center, lower left, upper right of the network. From the illustrative instance in Section 3.4.1, testing retained $\Psi = \{1, 2, 3\}$ asset types with $\Psi^D = \{1, 2\}$ and $\Psi^P = \{3\}$. It affixed $|K_1| = |K_2| = 2|K_3|$ as the number of assets of each type to represent the expected need for fewer assets providing proximal service (e.g., command-and-control assets), as described for the motivating civil scenario in Section 3.1. The origins for assets of types in Ψ^D are located in the upper left and lower right of the network, whereas the origins for assets of types Ψ^P are on the left and/or right sides of the network. Figure 13 depicts an example network for $|N| = 96$, $|N^d| = 5$, and $(|K_1|, |K_2|, |K_3|) = (4, 4, 2)$. For hexagonal meshes having the same number of hexagons in a column as the number of columns, this arrangement of demands vis-á-vis assets of different types and their disparate origins requires routing decisions that coordinate the sequence with which demands are serviced, the routing of assets along different paths, and the synchronization of their servicing of specific demands in non-trivial scenarios. Emulating the convention from Figure 12, the red stars within Figure 13 denote the demand nodes, and the purple diamonds, blue circles, and green triangles respectively indicate the origins of assets of type $\psi = 1, 2, 3$. This collective dispersion of demand nodes and originating asset locations represents the disparate basing of assets, ensures comparison of solution method performance across different network sizes is reasonable, and challenges the solution methods.

Test instance generation retained $c_{ijk} = 1,\ \forall\ (i, j) \in A, k \in K$ from the illustrative instance in Section 3.4.1, and it affixed $\tau_{ijk} = 0.8,\ \forall\ (i, j) \in A, k \in K_1 \cup K_3$ and $\tau_{ijk} =$

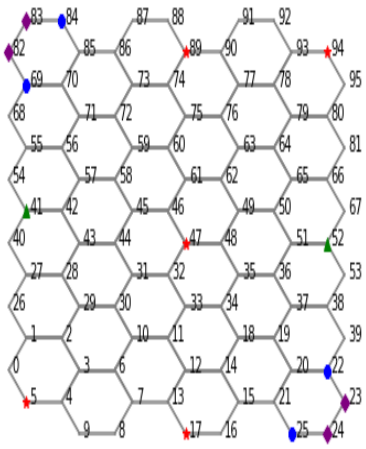


Figure 13. Example Test Network and Placement of Demand Nodes and Assets

1, $\forall\ (i,j) \in A, k \in K_2$. Each demand within an instance has the same designated service window width and earliest possible service time. Different such demand-specific parameters are certainly possible and are important for some applications; we use common initial values to simplify the generation of instances, avoid generating an infeasible CoVRP-PS instance (e.g., wherein a demand-specific service window would require an asset to slowly traverse more arcs than exist in the network), and enable a more direct comparison of alternative solution methods via this technique during computational testing in subsequent sections.

All testing was performed on an Intel(R) Core(TM) i7-10875H CPU @2.30GHz with 128 GB of RAM on a 64-bit operating system, and using Python (Version 3.9.7) with the GurobyPi package to invoke the commercial solver Gurobi (Version 9.5.1). Alternative termination criteria were 600 seconds (i.e., 10 minutes) of computational effort and an identified 0.0005% relative optimality gap.

Testing explored the six potentially significant problem factors by solving one instance at each of 162 combinations of factor levels, given three levels for each factor except for the number of asset routed. Testing considered such levels, routing either 5 or 10 total assets. Analysis of the results identified all but $\overline{LB}_i$ to be a significant

factor, with its p-value equal to 0.5292 and all others below 0.009. Of note, the levels of each factor were comparable to the recommended testing regime in Table 4, with two exceptions. First, Table 4 omits the levels for the factor determined to not be significant (i.e., $\overline{LB}_i = 0, 2, 5,\ \forall\ i \in N^d$). Second, this initial testing only explored $|N| = 48, 96, 160$.

As intimated, Table 4 presents the CoVRP-PS selected problem parameters and their respective levels for comparative testing of the nine alternative solution methods. Subsequent testing also considers a fourth level of $|N| = 240$ to challenges the solution methods. With respect to the $\overline{LB}_i$-values, all subsequent instances affix these at 2 because earlier service times are not attainable for any demand, in any instance because of the network topology and dispersion of both demands and assets. Subsequent testing examined 1944 combinations of nine alternative solution methods, each applied to the 216 instances corresponding to the parametric variations listed in Table 4.

Table 4. Varied Problem Parameters and Levels Explored for the Designed Experiment

Parameter	Levels Explored
$\|N\|$	$48, 96, 160, 240$
$\|N^d\|$	$1, 3, 5$
$(\|K_1\|, \|K_2\|, \|K_3\|)$	$(2, 2, 1), (4, 4, 2)$
δ	$1, 3, 5$
$\left(\overline{UB}_i - \overline{LB}_i\right),\ \forall\ i \in N^d$	$2, 6, 10$

3.4.3 Performance of Direct Optimization

Testing first examined Gurobi for solving CoVRP-PS instances directly, without the use of a model decomposition heuristic or a preprocessing technique. Tables 5 and 6 report the testing results for respective $(|K_1|, |K_2|, |K_3|)$-values of $(2, 2, 1)$ and $(4, 4, 2)$. The first row in each table indicates the δ-value, and the second row indicates the service time window $\overline{UB} - \overline{LB}$, indicated as without demand-node specific indexing. The first and second columns respectively denote the number of

nodes $|N|$ and the the number of demand nodes $|N^d|$. For every such parametric combination, Tables 5 and 6 report the runtime (i.e., RT, in seconds) and the relative optimality gap identified upon termination. An entry of an asterisk $*$ indicates Gurobi did not identify a feasible solution after 600 seconds of computational effort.

Gurobi was able to identify a feasible solution for 80.6% of instances. The smallest instances for which Gurobi consistently could not identify a feasible solution entailed routing 5 assets over a network with 160 nodes to service 5 demands. The solver had similar difficulties routing 5 assets to service only 3 demands over a network having 240 nodes. For the latter case, Gurobi was only able to identify a feasible solution for 3 instances comprised of parametric variants of δ and $(\overline{UB} - \overline{LB})$.

Over the 174 instances for which Gurobi did find a feasible solution, it required an average of 80.0 seconds of computational effort and identified solutions having an average relative optimality gap of 11.8%. Gurobi identified a global optimal solution for 101 instances (i.e., 46.8%) of the 216 tested. Focusing on the remaining 33.8% of instances, the average reported relative optimality gap was 28.1%.

Readily discernible in Tables 5 and 6, Gurobi struggled to find high quality solutions when solve instances with a large network size (i.e., $|N| \geq 96$) and a relatively higher number of demands (i.e., $|N^d| \geq 3$), more so with smaller service time windows. Larger $(\overline{UB} - \overline{LB})$-values did yield better solver performance. Such an outcome results from the existence of alternative optima when routing assets due to both alternative routes over a regular, tessellated mesh network and alternative timing for some assets on a given route that will service demands within the adjusted time window. However, a larger time service window may not be suitable for a CoVRP-PS application, depending on how nearly simultaneous the asset types' collaborative service of demands must be. By comparison, the δ-parameter did not exhibit a notable trend in outcomes.

Table 5. CoVRP-PS Testing Results for $(|K_1|, |K_2|, |K_3|) = (2, 2, 1)$

		$\delta = 1$						$\delta = 3$						$\delta = 5$					
$\overline{UB} - \overline{LB}$		2		6		10		2		6		10		2		6		10	
		RT	**Rel**	**RT**	**Rel**	**RT**	**Rel**	**RT**	**Rel**	**RT**	**Rel**	**RT**	**Rel**	**RT**	**Rel**	**RT**	**Rel**	**RT**	**Rel**
$\|N\|$	$\|N^d\|$	(sec)	**Gap**	(sec)	**Gap**	(sec)	**Gap**	(sec)	**Gap**	(sec)	**Gap**	(sec)	**Gap**	(sec)	**Gap**	(sec)	**Gap**	(sec)	**Gap**
	1	0.3	0	0.2	0	0.2	0	0.4	0	0.2	0	0.2	0	0.4	0	0.2	0	0.2	0
48	3	2.6	0	1.4	0	3.4	0	6.2	0	0.8	0	0.8	0	9.1	0	1.2	0	0.8	0
	5	15.3	0	1.3	0	0.5	0	10.5	0	4.3	0	0.5	0	18.4	0	4.2	0	0.6	0
	1	1.9	0	1.5	0	1.5	0	1.8	0	1.6	0	1.6	0	1.4	0	1.7	0	1.2	0
96	3	600.3	0.04	600.3	0.01	278.6	0	600.3	0.01	600.3	0.01	268.5	0	600.2	0.06	600.3	0.01	251.1	0
	5	600.1	0.14	189.8	0	101.6	0	*	*	208.9	0	53.4	0	600.3	0.14	600.4	0.07	202.5	0
	1	221.5	0	377.9	0	332.3	0	229.3	0	276.6	0	288.9	0	600.4	0.06	259.8	0	312.4	0
160	3	600.1	0.37	600.1	0.33	600.1	0.32	600.1	0.27	600.1	0.27	600.1	0.32	600.1	0.39	600.1	0.36	600.1	0.31
	5	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
	1	600.1	0.31	600.1	0.27	600.1	0.24	600.1	0.29	600.1	0.27	600.1	0.24	600.1	0.33	600.1	0.27	600.1	0.29
240	3	*	*	*	*	*	*	600.4	0.75	*	*	600.2	0.56	*	*	600.2	0.56	*	*
	5	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

* Gurobi terminated after 600 seconds without identifying a feasible solution

Table 6. CoVRP-PS Testing Results for $(|K_1|, |K_2|, |K_3|) = (4, 4, 2)$

		$\delta = 1$						$\delta = 3$						$\delta = 5$					
$\overline{UB}_i - \overline{LB}_i$		2		6		10		2		6		10		2		6		10	
		RT	**Rel**	**RT**	**Rel**	**RT**	**Rel**	**RT**	**Rel**	**RT**	**Rel**	**RT**	**Rel**	**RT**	**Rel**	**RT**	**Rel**	**RT**	**Rel**
$\|N\|$	$\|N^d\|$	(sec)	**Gap**	(sec)	**Gap**	(sec)	**Gap**	(sec)	**Gap**	(sec)	**Gap**	(sec)	**Gap**	(sec)	**Gap**	(sec)	**Gap**	(sec)	**Gap**
	1	0.3	0	0.3	0	0.3	0	0.5	0	0.3	0	0.3	0	0.9	0	0.3	0	0.3	0
48	3	3.5	0	1.0	0	0.9	0	14.0	0	1.0	0	1.1	0	331.2	0	1.1	0	1.0	0
	5	5.3	0	1.1	0	1.2	0	51.3	0	1.3	0	1.5	0	31.4	0	1.6	0	1.2	0
	1	3.8	0	3.4	0	3.3	0	12.4	0	3.4	0	3.0	0	5.0	0	4.2	0	4.1	0
96	3	600.2	0.01	600.1	0.01	393.7	0	600.1	0.06	409.9	0	245.9	0	600.1	0.13	343.3	0	301.4	0
	5	600.1	0.07	41.0	0	36.5	0	600.1	0.09	38.2	0	39.2	0	600.1	0.13	86.6	0	38.4	0
	1	600.1	0.11	600.2	0.11	600.2	0.11	600.1	0.18	545.4	0	600.1	0.07	600.2	0.21	592.1	0	600.2	0.15
160	3	600.1	0.36	600.2	0.38	600.1	0.33	600.1	0.40	600.2	0.39	600.1	0.36	600.2	0.42	600.1	0.40	600.1	0.38
	5	*	*	*	*	600.2	0.18	*	*	*	*	600.2	0.17	*	*	*	*	600.6	0.22
	1	600.1	0.37	600.1	0.40	600.1	0.34	600.1	0.42	600.1	0.34	600.2	0.37	600.2	0.40	600.2	0.34	600.1	0.37
240	3	*	*	600.5	0.55	600.2	0.56	600.2	0.57	600.2	0.55	600.3	0.56	*	*	601.1	0.55	600.8	0.53
	5	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

* Gurobi terminated after 600 seconds without identifying a feasible solution

In aggregate, Gurobi performed well, but it did struggle within the 600 second limit on computational effort to find high quality solutions and eventually to find feasible solutions as instance sizes grew, more so with low δ-values. We find this shortcoming when solving more challenging CoVRP-PS instances to merit the comparative analysis of alternative solution methods in the following section.

3.4.4 Comparative Testing Results on Tessellation Induced Networks

Comparing the nine different models explored directly allows for an examination of their relative efficacy. Of interest is not only the ability to find feasible solutions consistently within a bounded amount of computational effort, but also the relative quality of the solutions that are found. To simplify the presentation of results and corresponding discussions, Table 7 provides the abbreviations for the nine solution methods.

Table 7. Model Abbreviation Guide

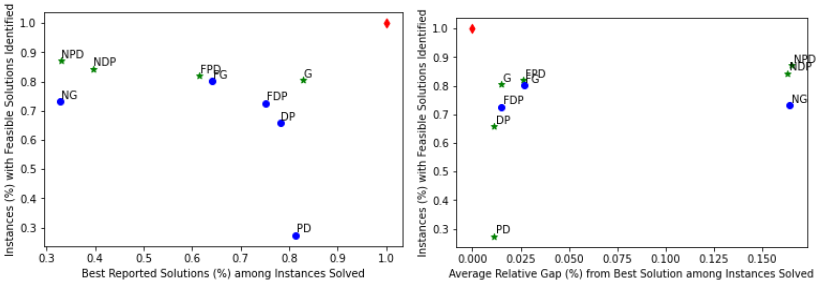
Model Decomposition	**Preprocessing Technique**		
(Ψ^1, Ψ^2)	None	Floyd-Warshall	Nearest Neighbor
$(\Psi, \emptyset)$*	G	FG	NG
(Ψ^D, Ψ^P)	DP	FDP	NDP
(Ψ^P, Ψ^D)	PD	FPD	NPD

* Indicates no model decomposition

Whereas G (i.e., solving an instance directly with Gurobi), FG, and NG terminated after 600 seconds of computational effort if an optimal solution was not identified, all other solution methods allotted 300 seconds to each of the subsequent optimization problems for the model decomposition heuristic to provide an equitable amount of time to solve the instances. The preprocessing techniques were quick, requiring no more than 6 seconds for the largest test instance, so testing did not impose a time limit on either of them.

Figure 14 presents the aggregate results of applying the nine combinations of model

decomposition heuristic and preprocessing technique to the 216 test instances. The location of a solution method in each subfigure corresponds to its performance with respect to two metrics. In Figure 14a, the vertical axis indicates the percentage of instances for which a solution method identified a feasible solution within 600 seconds, whereas the horizontal axis indicates, for those instances, the percentage of which that solution method identified the best reported solution among the nine solution methods. Within Figure 14a, the red diamond denotes the ideal point (Ehrgott, 2008): identifying a feasible solution for every instance and it always being the best one found. Either a blue circle or a green star indicates the performance for each of the nine solution methods. The green star indicates the performance is Pareto optimal; no other method dominates its performance with respect to both metrics. All Pareto-optimal models are considered equally good, given a set of valued performance metrics (Bazaraa et al., 2008). In Figure 14b, the vertical axis is the same as Figure 14a, and the horizontal axis indicates a different solution quality metric: the average relative gap (%) for identified feasible solutions, as compared to the best solutions identified by all solution methods. This latter metric helps identify a solution method that may not often find the best solution but consistently finds (relatively) high quality solutions. The red diamond again indicates an ideal point, and the green stars denote Pareto optimal performance by solution methods.



(a) Best Reported Solutions (%) **(b) Average Relative Gap (%) from Best Solution Found**

Figure 14. Average Performance of 9 Solution Methods over All Test Instances

Within Figure 14a, G is Pareto optimal. Although it did not find a feasible solution for the greatest percentage of instances, it did find the best solution for the highest percentage of instances for which it did find a feasible solution. The tradeoff in performance metrics is non-trivial. To find a feasible solution to 5-10% more of the instances (e.g., via FPD, NDP, or NPD) would require a sacrifice, dropping to those solutions being the best identified only 62% or possibly as low as 33% of the time.

Figure 14b better illustrates the performance of FPD. Whereas G solutions were within 1.5% of the best reported, on average, the FPD solutions were within 2.7%. In contrast, NDP and NPD did not do well in Figure 14b; their identified solutions had a 16% relative gap from the best reported solutions, on average. DP and PD were also Pareto optimal in Figure 14b, but their relatively poor performance for identifying solutions is challenging to endorse.

In terms of the algorithmic components, several trends are identifiable in Figures 14. Either permutation of the model decomposition heuristic in the absence of a preprocessing technique yielded the worst two performances, in terms of finding feasible solutions. When combined with the Floyd-Warshall preprocessing technique, the decomposition heuristic performance identified more feasible solutions with a marginal degradation in solution quality. By comparison, both model decomposition permutations using the Nearest Neighbor preprocessing technique found more solutions overall, but with notably lesser solution quality.

Such a relative result between the preprocessing techniques when augmenting either DP or PD makes sense. The Floyd-Warshall technique applies a very simple lower bound on the time that must be allotted for the subsequent routing of asset types Ψ^2 when routing asset types Ψ^1. Assuming every demand is serviced as quickly as possible (i.e., as a first priority) in the second stage, the α_i^{LB}-values in FDP and FPD do not eliminate an optimal solution to the original problem. Therefore, any

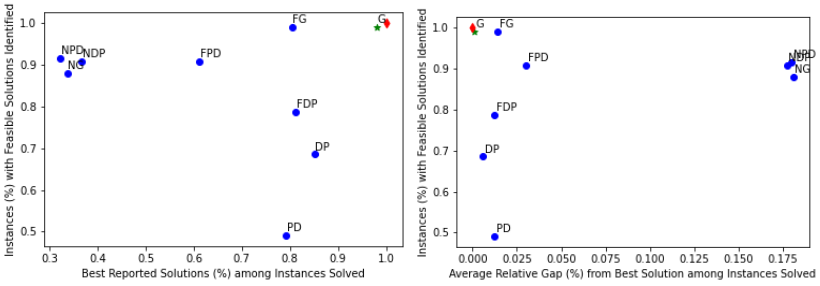
lesser solution quality for these solution methods resulted from an inability to solve either or both stage-specific problems optimally within 300 seconds.

In contrast, the Nearest Neighbor technique applies a more elegant approximation of α_i^{LB}-values that considers a sequence of servicing demands by asset types Ψ^2. This construct explains the relatively greater ability of NDP and NPD for identifying feasible solutions. Of note, neither does so for all instances, due in part to routing different numbers of instances for each type of asset across different stages of the model decomposition heuristic. Moreover, these cuts may eliminate an optimal solution because, as a heuristic, the Nearest Neighbor algorithm may overestimate α_i^{LB}-values. Both the higher number of feasible solutions identified and the relatively lesser quality solutions of NDP and NPD are attributable to such cuts.

Meanwhile, neither preprocessing technique enhanced Gurobi's average performance when solving CoVRP-PS instances directly. With respect to the set of solution methods, neither FG nor NG demonstrated Pareto optimal performance.

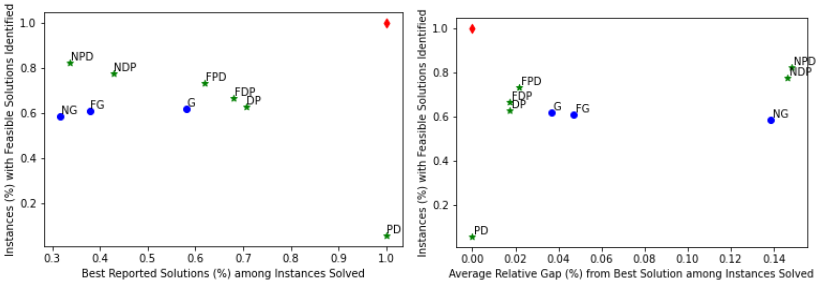
Relative Performance by Network Size Within Section 3.4.3, the number of nodes $|N|$ challenged Gurobi notably when $|N| \geq 160$. As such, it is important to examine the relative performance of the solution methods with greater nuance. Figure 15 divides the instances into two sets of 108 instances each: $|N| \in \{48, 96\}$ (i.e., "small grids") and $|N| \in \{160, 240\}$ (i.e., "large grids"). Using the same format for presenting results as Figure 14, Figures 15a & 15b and Figures 15c & 15d respectively illustrate the solution methods' relative average performance for small grid and large grid network instances.

Figures 15a & 15b show that G exhibited excellent performance when solving small grid instances by identifying feasible solutions for 99.1% of instances and, among them, finding the best solution 98.1% of the time. The only other model that appeared near Pareto optimal solution method for small grid instances was FG, enhancing a di-



(a) Best Reported Solutions (%), $|N| \in \{48, 96\}$

(b) Average Relative Gap (%) from Best Solution Found, $|N| \in \{48, 96\}$



(c) Best Reported Solutions (%), $|N| \in \{160, 240\}$

(d) Average Relative Gap (%) from Best Solution Found, $|N| \in \{160, 240\}$

Figure 15. Model Comparison Based on Number of Nodes $|N|$

rect solution of instances with the Floyd-Warshall preprocessing technique. However, identifying the same number of feasible solution was accompanied by moderately and marginally worse performance for the respective solution quality metrics in Figures 15a & 15b.

Other trends for all instances were manifest for small grid instance performances. Model decomposition methods without preprocessing techniques identified the fewest feasible solutions. The Floyd-Warshall technique improved that performance with minor degradation in solution quality, and the Nearest Neighbor technique helped model decomposition methods find yet more feasible solutions but with a more notable decrease in average solution quality.

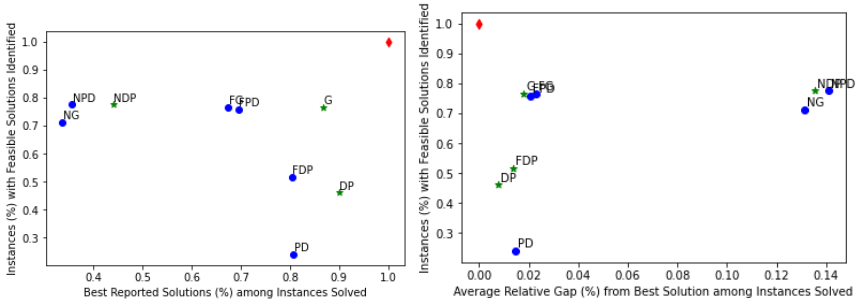
For large grid instances, Figures 15c and 15d exhibit compelling results. Of note, G is not Pareto optimal. Moreover, the only solution methods that are not Pareto optimal are the ones that do not leverage either permutation of the model decompo-

sition heuristic. Relative trends among the model decomposition heuristic variants remain consistent with previous observations. PD and DP find the least feasible instances among solution methods on the Pareto front. Enhancing them with the Floyd-Warshall preprocessing technique identifies more feasible solutions with marginally worse average solution quality. Enhancing them with the Nearest Neighbor preprocessing technique found the most feasible solutions ($\sim 80\%$) with a disconcerting cost to solution quality (i.e., within $\sim 15\%$ of the best solution identified, on average).

In aggregate, one may reasonable conclude that a model decomposition heuristic should be used for instances of CoVRP-PS having a large number of nodes in the network. Depending on one's priority between identifying feasible solutions and high quality solutions, the heuristic should be enhanced with a preprocessing technique. Considering the results in Figure 15c and 15d, we would recommend using the Floyd-Warshall preprocessing technique (i.e., either FDP or FPD) based on their relative proximity to the ideal points of performance.

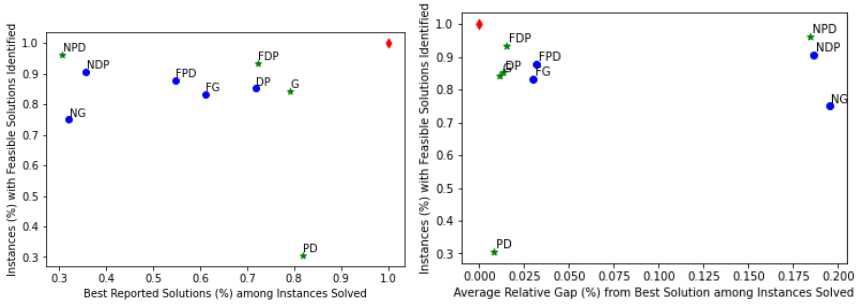
Relative Performance by Number of Assets Routed Also noted in Section 3.4.4 was a difference in Gurobi's performance for G when routing either five or ten total assets (i.e., with fixed relative numbers of assets, by type). Figure 16 presents the comparative results of the nine solution methods, partitioned by the number of assets routed. Figures 16a & 16b and Figures 16c & 16d present the solution methods' relative average performance for 108 instances each of routing $(|K_1|, |K_2|, |K_3|) = (2, 2, 1)$ and $(|K_1|, |K_2|, |K_3|) = (4, 4, 2)$ assets.

G is Pareto optimal in every performance comparison within Figure 16. Such a result implies its general efficacy, but the results in Figures 15c and 15d indicated otherwise; G does not perform as well as other methods for larger network sizes. Within that context, its good performance without regard to the number of assets routed merely indicates this problem parameter is not useful for distinguishing when



(a) Best Reported Solutions (%), $(|K_1|, |K_2|, |K_3|) = (2, 2, 1)$

(b) Average Relative Gap (%) from Best Solution Found, $(|K_1|, |K_2|, |K_3|) = (2, 2, 1)$



(c) Best Reported Solutions (%), $(|K_1|, |K_2|, |K_3|) = (4, 4, 2)$

(d) Average Relative Gap (%) from Best Solution Found, $(|K_1|, |K_2|, |K_3|) = (4, 4, 2)$

Figure 16. Model Comparison Based on Number of Assets Routed

to set aside G in favor of an alternative solution method.

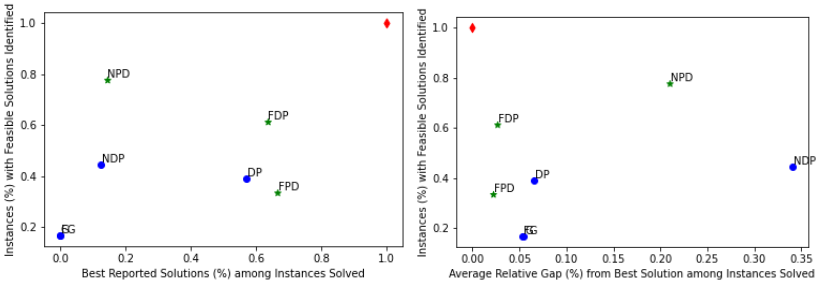
When routing five assets, Figures 16a and 16b exhibit previously noted trends. The two permutations of the model decomposition heuristic without a prepossessing technique find the least number of feasible solutions. The Floyd-Warshall preprocessing technique enhances that result with marginally degraded average solution quality, and the Nearest Neighbor preprocessing technique helps the model decomposition heuristic find a few more feasible solutions with notably worse average solution quality. Meanwhile, augmenting G with a preprocessing technique does not help.

When routing more assets, the trends in Figures 16c and 16d differ. Only PD performs poorly with respect to identifying feasible solutions, yet it is still Pareto optimal because of the relative quality of those solutions, via either performance

metric. All other variants of the model decomposition heuristic, with-or-without preprocessing techniques, identify feasible solutions for over 90% of instances, with Pareto optimal performance by FDP and NPD for both solution quality metrics.

Relative Performance on the Most Challenging Instances Alternative examinations of average solution method performance by the different combinations of δ and $\overline{UB_i} - \overline{LB_i}$ exhibited relative solution method performance consistent with previously observed trends. Accordingly, we forgo displaying those results in favor of brevity. As a notable exception, FPD yielded the only Pareto optimal performance for $\delta = 5$ and $\overline{UB_i} - \overline{LB_i} = 10$.

Instead, a final examination compared solution method performance on only the 18 instances we identified as most challenging to G: $|N| \in \{160, 240\}$, $|N^d| = 5$, and $(|K_1|, |K_2|, |K_3|) = (4, 4, 2)$. Figure 17 presents the average performance of the nine solution methods. Of note, neither PD nor NG appear in Figure 17a or 17b; neither solution method identified a feasible solution for any of the 18 instances. Also, G and FG performed equivalently, and they are depicted at the same locations in each of Figures 17a and 17b.



(a) Best Reported Solutions (%) **(b) Average Relative Gap (%) from Best Solution Found**

Figure 17. Most Challenging Instances Explored

For these most challenging instances, G is not Pareto optimal. It only found a feasible solution to 17% of the instances and, although it never found the best

solutions, it was within 6% of the best solutions identified, on average.

Both permutations of the model decomposition method when augmented with the Floyd-Warshall preprocessing technique were Pareto optimal, as was the routing of proximal-servicing assets first when augmented with the Nearest Neighbor preprocessing technique. As with other comparisons, the FDP and FPD techniques merit consideration in parametric regimes of CoVRP-PS that challenge G when a balance is desired between the likelihood of identifying a feasible solution and the solution quality attained. To maximize the likelihood of identifying a feasible solution, NPD is recommended, acknowledging that solution quality may not be relatively worse. As computing resources allow, a parallel implementation of G, FDP, FPD, and NPD is preferable to identify high-quality, feasible solutions to CoVRP-PS instances in challenging parametric regimes.

3.5 Conclusions and Recommendations

For a new subclass of the vehicle routing problem, this research set forth a mathematical programming model to route assets over a network and service demands, subject to complicating problem factors. These factors included the collaborative servicing of demands by different asset types, which we modelled via adjustable time windows to enforce nearly simultaneous servicing, and the ability of a subset of asset types to provide service proximally.

Initial testing identified the problem parameters that significantly challenge the identification of high-quality, feasible solutions by a leading commercial solver. Those parameters are the number of nodes in the network; the number of demand nodes to be serviced; the number of assets to be routed; the service window width, as represented by the imposed time window; and the increment with which that window may be shifted an integer-valued number of times. Initial testing likewise informed

the development of both two permutations of a model decomposition heuristic and two preprocessing techniques designed to improve their respective performance.

A designed experiment enabled comparative testing of nine resulting solution methods over a set of 216 test instances. Testing revealed the parametric regimes in which direct optimization performed poorly, and it identified via Pareto front analysis the solution methods having the greatest merit. When routing fewer assets to service many demands over larger networks, the most feasible solutions were identified by first routing asset types that provide proximal service, but doing so by imposing lower bounds on demand service windows, as informed by a Nearest Neighbor preprocessing technique. This solution method does degrade relative solution quality, so viable alternatives include bounds informed by a preprocessing technique based on the Floyd-Warshall algorithm, combined by either permutation of the model decomposition heuristic. Of course, direct optimization via a commercial solver remains preferable for easier-to-solve instances.

A natural extension to this work should examine yet larger problems that will challenge every solution method examined. Such challenges imply the need to apply a problem decomposition approach that partitions the network, the demands, and/or the assets to be routed into smaller CoVRP-PS instances. Conceptually simple, the implementation of such a problem decomposition merits creative thought and extensive computational evaluation. Of interest are the means by which the problem can be decomposed and the relative efficacy of the resulting solutions attained via either direct optimization or one of the better performing alternative solution methods demonstrated in this research.

IV. A Stackelberg Framework for Disrupting Coordinated, Multi-asset Routing and Sequential Servicing of Demands

This research explores the problem of routing different types of assets having disparate but complementary capabilities (or commodities to deliver) over a directed network to service a set of demands sequentially. Given n asset types, an asset of each type must visit a demand to service it, with Type 1 visiting no later than Type 2, Type 2 no later than Type 3, et cetera, and where the time between first and last asset arrival is bounded. The network is also subject to outside interference; actions may slow down asset travel on a subset of arcs within the network, delaying the servicing of demands. Of interest is how to identify where such interdiction actions are most effective. From one perspective, such locations are opportunities to slow deliveries. From the alternative perspective, such locations indicate potential network vulnerabilities to address either preemptively or with mitigating actions.

Consider the following motivating civil scenarios. Commodities are being transported to major construction efforts across a region. They require delivery in timely manners to various sites, and their delivery sequence is relevant (e.g., concrete before framing material, then electrical conduits and outlets, after which the wallboard arrives). Traffic congestion can slow traffic on portions of potential delivery routes. Such delays can disrupt scheduled deliveries. Yet, knowing the locations where delays are most impactful and rerouting is not helpful will inform where mitigating actions (e.g., coordinated security escort) have merit.

Similarly, after a natural disaster or heavy storm, some capabilities or commodities should be delivered as assistance to impacted towns before others (e.g., rescue equipment and medical assistance are needed before building repair materials). Road network conditions may be degraded, making travel over certain road segments take more time. Understanding where road damage would be most impactful can inform

decisions regarding where to stage a limited road repair capability.

Alternatively, consider a related military scenario wherein an adversary has intelligence regarding the location of several assets they seek to destroy. First, intelligence, surveillance, and reconnaissance (ISR) assets must validate current intelligence. Then, strike assets (e.g., aircraft, vehicles), move to the locations and deliver munitions to destroy the targets. Such a process to "service demands" is also necessarily sequential in nature. In a defensive posture, we seek to conduct localized electronic interference, an aspect of an anti-access/area denial (A2/AD) environment (Alcazar, 2012; Neagoe and Borşa, 2019). This communication interference can increase travel time in targeted areas of a network, as up-to-date information cannot be refreshed or confirmed. The desired effect of these actions is to delay the adversary's attacks and provide time to move or protect their targets from destruction. Thus, it is of interest to identify where to apply electronic interference most effectively.

These realistic scenarios accentuate the importance of this research, which identifies a model and accompanying solution method to address the following problem statement, as framed for the later, military perspective, but which is valid for either category of application.

Problem Statement Given a limited budget for arc-specific disruptions, each of which increases arc travel times, we seek a network interdiction strategy to maximally delay the cumulative servicing of demands by an adversary having assets that traverse a network to service multiple demands at different, fixed locations in a coordinated, sequential manner.

Within this statement, an interdiction strategy consists of actions to disrupt the flow of assets over the network by delaying them. Hereafter, we use the terms *interdiction*, *disrupting action*, and *delaying action* interchangeably.

4.0.1 Literature Review

Informative to this research is the literature on bilevel programming, network interdiction modeling, and corresponding solution methods.

Bilevel Programming Frequently in the literature, one can find the problem of attacking a network paired with the defense of the same network, represented via a bilevel program model. These attacker-defender models are a type of Stackelberg game (Stackelberg, 1952), where the attacker and defender's actions are sequential. Given an attack that degrades a network, how will the defender best achieve their objectives? Given that best response, what attack is most effective? Bilevel programming models were originally proposed to model such Stackelberg games that appear so often in leader-follower games in the market economy (Dempe, 2002). Such decision-maker objectives compete while subject to their respective set of interdependent constraints. Applications are seen in highway networks with objectives modeling operating costs, travel time functions, accident costs, and maintenance repair expenses (Bard, 2013). The preliminaries of the model used in this research assume that the participants do not exchange information, making cooperation prohibited. Due to the problem's non-cooperative nature, participants cannot negotiate, introducing non-Pareto optimal solutions (Moore and Bard, 1990). The top level participant is assumed to anticipate the reactions of the lower-level problem and seeks to identify an optimal strategy accordingly (Colson et al., 2007).

The competitive framework influences the solution methods of a bilevel program in the problem definition. Many classical bilevel interdiction problems model a zero-sum game wherein participants compete over a common objective; the upper-level decision-maker seeks to maximize (or minimize) the minimum (or maximum) value the lower-level decision-maker can achieve (Smith and Song, 2020). Wood (1993) pre-

sented a problem wherein one participant seeks to maximize flow on the network. The interdictor attempts to minimize the maximum flow by interdicting arcs and limiting the resource flow *a priori*. This type of problem scenario is most similar to the work presented in this research, where participants compete over an objective related to travel time on the network. Bilevel programs also model non-zero-sum games wherein the decision-makers have different objective functions. (Dempe et al., 2005) studied a math programming model for natural gas shippers that attempted to maximize revenue while minimizing the number of transactions. (Ma et al., 2022) investigated energy outputs, where the upper-level model maximizes the benefits of sharing energy storage with a lower-level mode to minimize system operating costs. Expanding these problems to have multi-objective formulation becomes more nuanced, expanding the problem complexity. For example, (Zhang et al., 2022) formulated a bilevel programming model with multiple objectives in the upper-level problem, attempting to minimize total cost and service tardiness of electric vehicle charging stations, and with a lower-level objective of minimizing total travel time between stations.

Multi-level programming models have a variety of objective functions that vary depending on the network problem, usually dependent on the defender. Network defender problems in the literature address maximizing flow (Wood, 1993), minimizing shortest path lengths and cost network flow (Israeli and Wood, 2002), and maximizing the probability of evasion (Lunday and Sherali, 2012*b*). (Starita and Scaparra, 2021) explored an attacker-defender bilevel program wherein an attacker destroyed arcs, seeking to maximize congestion in a user equilibrium state. Other network interdiction problems more closely align with the problem statement set forth by this research, investigating the Vehicle Routing Problem (VRP) while accounting for time. (Sadati et al., 2020*a*) modeled an attacker-defender depot-interdiction model. In a sequel, (Sadati et al., 2020*b*) modeled a defender-attacker-defender trilevel program with a

VRP in the lowest level. It represented defender protection of depots prior to the depot interdiction and operation, with competition over composite objective function related to operating costs, travel costs, and unsatisfied demands.

Aside from a few works explicitly using time in network interdiction models as a measure of loss of efficiency on asset flow, our review of the literature identified no research that examines interdiction of multiple assets having relationships as in the underlying problem, i.e., wherein sequential servicing of demands by different asset types must occur in an order and within a limited duration of time. Moreover, where most previous studies explored arc-wise network interdiction to render arcs unusable, this research allows the use of the affected arcs with increased traversal times as a consequence.

Network Interdiction Models Network interdiction models are prevalent in the literature regarding application and solution methods (e.g., see (Smith and Song, 2020)). Bilevel network interdiction models come in the form of defender-attacker and attacker-defender models, where the actions of one party occur first, causing the second party to respond. The underlying problem studied herein could alternatively be categorized via either framework. For civil applications, interference with the network could be characterized as an attack, and the decision-maker operating on the network would be the defender. For military applications, the transit of multiple asset types over the network may be an attack by an adversary, and preliminary actions to slow travel would be defensive in nature. For simplicity of discussion, the remainder of this paper frames the problem as using only one such lens. Hereafter, we embrace the defender-attacker framework for its conceptual virtue, i.e., degrading an adversary's ability to cause harm by deliberately slowing down their assets travel in subsets of the network. Thus, network interdiction is framed as a defensive action.

Network modeling in the form of defender-attacker often tries to prevent an attack

or degradation of a system or at least minimize a metric of loss. For example, in research by Lei et al. (2018), an attacker tries to destroy a subset of arcs on a network that will minimize the system's source-to-sink flow by randomly interdicting an arc with a measure of uncertainty. The study investigates the defender's choice to increase arc capacity to mitigate loss after the attacker's plan has been decided. Risk-averse and risk-neutral behaviors of the attacker and defender are investigated (Lei et al., 2018). Perea and Puerto (2013) proposed a math programming model that optimizes a network's allocation of resources over an existing railway system to minimize the negative consequences of an attack. Not all defensive models focus on a physical attack nor degradation of the system. For example, Zokaee et al. (2016) considered a humanitarian logistics network for disaster relief operations by modeling the shortages of relief commodities flowing through the system. The model investigates a three-level chain model of suppliers, distribution centers, and affected areas considered, seeking to maximize the satisfaction level of the affected population.

As intimated previously, some literature frames preliminary actions on a network followed by operations thereon as an attacker-defender framework. Insight into such attacker-defender models, and even the trilevel variant of attacker-defender-attacker models, reveals an advantage to the attacker every time. The defender is tasked with protecting an extensive network with limited assets to minimize disruption. In contrast, the attacker need only attack a subsection of the network (Brown et al., 2006). Verter and Dasci (2002) modeled the simultaneous optimization of plant locations with capacities on acquisition and technology selection decisions in a multi-commodity environment. Jouzdani et al. (2013) combined asset location and network flow considerations by minimizing the cost of facility locations, traffic congestion, and transportation of milk and dairy products under uncertain conditions. The authors' model utilized periods of demand uncertainty in a planning horizon to determine

optimal facility location and production volumes (Jouzdani et al., 2013).

Solution Methods Bilevel programming problems are challenging to solve; even simple linear bilevel programs are $\mathcal{NP}$-hard (Jeroslow, 1985). Under certain circumstances, a bilevel program can be reformulated as a single-level math program and solved directly. Such cases occur when the lower-level problem is a convex optimization problem; reformulation of zero-sum games occurs by taking the dual of the lower-level problem (Wood, 1993), and of non-zero-sum games by replacing the lower-level problem with its Karush-Kuhn-Tucker necessary optimality conditions (Colson et al., 2007). These methods are likewise suitable for lower-level problems having integer-restricted variables, if certain conditions apply to the integer-relaxed variant (e.g., total unimodularity of a linear constraint matrix).

In the absence of such properties, bilevel programs with integer-restricted lower-level decision variables are more difficult to solve. If the upper-level decision-variables are integer- or binary-restricted, some optimal approaches exist. (Bard and Moore, 1992) converted a two-level problem into a single-level problem, iteratively solving it via a branch-and-bound technique. Branch and bound performance has improved by applying plane cutting techniques (DeNegre and Ralphs, 2009). Complementary pivoting has also proven effective in finding global optimal solutions (Bialas and Karwan, 1984).

If the upper-level decision variables are not integer-restricted (Robbins and Lunday, 2016) or if they are but the problem is more challenging (e.g., nonlinear or large problem instances), a popular approach is to apply heuristics or metaheuristics to solve the upper-level problem while seeking the corresponding best solution(s) in the lower-level. These methods often include using trust-region methods (Conn et al., 2000) and penalty function methods (Ishizuka and Aiyoshi, 1992). Other techniques include simulated annealing (Sahin and Ciric, 1998), genetic algorithms (Yin, 2000;

Calvete et al., 2008), and particle swarm optimization (Wang et al., 2007), as well as hybrid approaches that combine these techniques (Kuo et al., 2015).

Section 4.1 sets forth the bilevel programming model to represent the motivating problem. Thereafter, it discusses the model's properties within the context of the aforementioned solution methods, informing two alternative approaches to solve problem instances.

4.0.2 Statement of Contributions

This paper makes two contributions to the network interdiction literature. First, it develops and validates a bilevel mixed-integer program for the problem of interest. A (lower-level) decision-maker routes assets of different types to sequentially visit demand nodes within limited duration time windows while minimizing the cumulative service times. In direct opposition, an (upper-level) decision-maker disrupts the network with actions to increase travel time on subsets of arcs, maximizing the minimum cumulative service times. Second, it proposes greedy and simulated annealing founded metaheuristic methodologies to find and improve feasible solutions to the Stackelberg model consistently.

Within the remainder of this paper, Section 4.1 presents the model and accompanying solution methodology. Section 4.2 illustrates the results of applying the solution method to a small, representative instance, scopes the computational experiments to test the relative efficacy and efficiency of alternative solution methods, reports the testing results, and discusses resulting insights. Thereafter, Section 4.3 concludes the research and suggests meaningful extensions.

4.1 Model Formulation and Solution Methodology

This section sets forth the model formulations in Section 4.1.1 and corresponding solution methodologies in Section 4.1.2 that will be evaluated in Section 4.2.

4.1.1 Model Formulation

Several assumptions are necessary to frame the model. First, the defender-attacker model is assumed to entail sequential decisions, wherein the defender interdicts the network and the attacker subsequently routes assets of different types to service demands in a coordinated manner. Second, the attacker is aware of the defender's interdiction strategy prior to routing assets. These two assumptions correspond to an extensive form game theoretic framework (i.e., a Stackelberg game) with perfect information, and the latter allows us to consider the attacker's best response to a given defender's strategy. Third, the defender has complete information about the attacker's asset routing and servicing problem. Such an assumption is appropriate as the defender has an excellent intelligence gathering enterprise. (For the alternative modeling perspective supporting civil applications, this assumption is also appropriate when seeking to identify the most effective disruptions, i.e., the greatest network vulnerabilities.) This assumption is relevant because it allows the defender to consider the attacker's best response to a given strategy and, in doing so, seek to identify the best interdiction strategy. Of note, because the defender seeks to maximize the cumulative service time of demands, which the attacker seeks to minimize, our bilevel programming model represents the interaction of a zero-sum, extensive form game theoretic framework with complete and perfect information (Shoham and Leyton-Brown, 2008).

The following sets, parameters, and decision variables make up the bilevel programming model studied through this paper. The model consists of multiple asset

types routed on a directed network to supply nodes from predetermined demand nodes. Different asset types are required to satisfy demand in a prearranged sequential order, and asset types are capable of traveling arcs at different speeds.

Sets

- $N = \{1, ..., n\}$: set of nodes in the transportation network or, alternatively, the set of nodes in a network induced by a regular tessellation to approximate continuous space, indexed alternatively on either i or j
  - $N^D \subset N$: subset of nodes corresponding to demand nodes
- A: set of directed arcs that connect nodes in a transportation network or, as examined for instances in Section 4.2 of this paper, in a regular tessellation that discretizes continuous space, indexed by (i, j)
- $G(N, A)$: the directed network
- $L = \{1, ..., l\}$: set of possible locations for a defender to conduct disruptive actions, each of which will slow down attacker travel on nearby arcs in the distribution network
- $\Psi = \{1, ..., |\Psi|\}$: asset types utilized to service demands, indexed on ψ, where ψ indicates the respective sequence in which at least one asset of the different asset types must individually visit a demand to collectively service it
- K: the set of assets, indexed on k and partitioned on Ψ, where $K_\psi \subset K \; \forall \; \psi \in \Psi$, $\bigcup_{\psi \in \Psi} K_\psi = K$, and $\bigcap_{\psi \in \Psi} K_\psi = \emptyset$

Parameters

- σ: a positive parameter indicating the additional amount of time an attacker's asset requires to traverse an arc due to *each* disruption $\ell \in L$ that affects the arc. Assumed via this parameter is that the cumulative effect of multiple disruptive actions is additive.

- η: the maximum number of delaying actions imposed by the defender that may be implemented against all locations $\ell \in L$

- $\alpha_{ij\ell}$: binary parameter equal to 1 if a delaying action at location ℓ will slow down defender travel on arc (i, j), and 0 otherwise. This research assumes arcs, where both nodes are within a bounded Euclidean distance of $\ell \in L$ are affected.

- b_{ik}: binary parameter equal to 1 if asset k is initially located at node i, and 0 otherwise.

- $s_k = \underset{i \in N}{\text{argmax}}\{b_{ik}\}$: the node at which asset k is initially located

- $\tilde{\tau}_{ijk}$: positive parameter indicating the length of time required to transverse arc (i, j) for asset k in the absence of interdiction

- ω: a scalar representing how much slower an agent may traverse an arc. For example, if $\omega = 2$, an agent may take twice as long as their fastest time to traverse an arc, i.e., their slowest speed is $1/\omega$ times their fastest speed.

- δ: parameter indicating the maximum range of time between when an asset of type $\psi = 1$ visits a demand until asset type $\psi = |\Psi|$ must visit the demand to collectively service it

Decision Variables

- y_ℓ: a non-negative integer-valued defender decision variable equal to the number of disruptive actions at location ℓ. Assumed via the integer-nature of this deci-

sion variable is that multiple disruptive actions may occur at the same location, and their cumulative effects on nearby arcs are additive.

- τ_{ijk}: positive decision variable indicating the length of time required to traverse arc (i, j) for asset k in the presence of interdiction
- x_{ijk}: binary decision variable equal to 1 if asset k traverses arc (i, j), and 0 otherwise
- u_{ik}: non-negative decision variable specific to each node i and asset k, used within the formulation to implement a lifted variant (Desrochers and Laporte, 1991) of Miller-Tucker-Zemlin (MTZ) subtour elimination constraints (Miller et al., 1960)
- t_{ik}: non-negative decision variable indicating the time at which asset k arrives at node i
- $\pi_{i\psi}$: non-negative decision variable indicating the time at which an asset of type $\psi \in \Psi$ visits demand $i \in N^D$
- γ_{ik}: binary decision variable equal to 1 if asset $k \in K_\psi$ visits demand $i \in N^D$ for all asset types $\psi \in \Psi$, and 0 otherwise

Given this notation, we formulate the bilevel problem **P** corresponding to this Stackelberg game as follows:

$$\mathbf{P}: \max_{\boldsymbol{y}} \min_{\boldsymbol{x}} \quad \sum_{i \in N^D} \pi_{i|\Psi|} \tag{35}$$

$$\text{s.t.} \quad \tau_{ijk} = \tilde{\tau}_{ijk} + \sigma \sum_{\ell \in L} \alpha_{ij\ell} y_\ell, \ \forall \ (i, j) \in A, k \in K, \tag{36}$$

$$\sum_{\ell \in L} y_\ell = \eta, \tag{37}$$

$$y_\ell \in \mathbb{Z}_+, \ \forall \ \ell \in L, \tag{38}$$

$$\sum_{j:(i,j)\in A} x_{ijk} - \sum_{j:(j,i)\in A} x_{jik} \leq b_{ik}, \ \forall \ i \in N, k \in K, \tag{39}$$

$$u_{ik} - u_{jk} + (|N| - 1)x_{ijk} + (|N| - 3)x_{jik} \leq |N| - 2, ...$$

$$... \ \forall \ (i,j) \in A | j \neq s_k, k \in K, \tag{40}$$

$$\sum_{j:(i,j)\in A} \sum_{k\in K_\psi} x_{ijk} \geq 1, \ \forall \ i \in N^D, \psi \in \Psi, \tag{41}$$

$$t_{ik} + \tau_{ijk} \leq t_{jk} + M(1 - x_{ijk}), \ \forall \ (i,j) \in A, k \in K, \tag{42}$$

$$t_{jk} \leq t_{ik} + \omega\tau_{ijk} + M(1 - x_{ijk}), \ \forall \ (i,j) \in A, k \in K, \tag{43}$$

$$t_{ik} \leq M \sum_{j:(j,i)\in A} x_{jik}, \ \forall i \in N, k \in K, \tag{44}$$

$$\pi_{i\psi} \geq t_{ik}, \ \forall \ i \in N^D, \psi \in \Psi, k \in K_\psi, \tag{45}$$

$$\pi_{i\psi} \leq t_{ik} + M(1 - \gamma_{ik}), \ \forall \ i \in N^D, \psi \in \Psi, k \in K_\psi, \tag{46}$$

$$\sum_{k\in K_\psi} \gamma_{ik} = 1, \ \forall \ i \in N^D, \psi \in \Psi, \tag{47}$$

$$\pi_{i(\psi-1)} \leq \pi_{i\psi}, \ \forall \ i \in N^D, \psi \in \Psi \setminus \{1\}, \tag{48}$$

$$\pi_{i|\Psi|} - \pi_{i1} \leq \delta, \ \forall \ i \in N^D, \tag{49}$$

$$x_{ijk} \in \{0,1\}, \ \forall \ (i,j) \in A, k \in K, \tag{50}$$

$$u_{ik} \geq 0, \ \forall \ i \in N, k \in K, \tag{51}$$

$$t_{ik} \geq 0, \ \forall \ i \in N, k \in K, \tag{52}$$

$$\pi_{i\psi} \geq 0, \ \forall \ i \in N^D, \psi \in \Psi, \tag{53}$$

$$\gamma_{ik} \in \{0,1\}, \ \forall \ i \in N^D, \psi \in \Psi, k \in K_\psi. \tag{54}$$

As indicated within the objective function (35), the defender and attacker respectively seek to maximize and minimize the cumulative time to collectively service

the demands. Constraint (36) computes the arc travel times τ_{ijk} as affected by the defender's interdiction strategy. The adopted representation is additive and linear. Alternative representations are possible, e.g., wherein disruptive actions impose a percentage increase to travel times, and may be embraced as appropriate for the applied problem of interest. Constraint (37) bounds the number of disruptive actions to be equal to η, and Constraint (38) limits the y_ℓ-variables to be non-negative and integer-valued, allowing more than one disruption at a given location.

For the attacker's routing problem, Constraint (39) enforces the conservation of flow for the movement of assets without requiring their return to respective origins. Constraint (40) applies a lifted variant of the MTZ subtour elimination constraints. Constraint (41) requires at least one asset k of each type $\psi \in \Psi$ visit each demand node $i \in N^D$.

Constraints (42) and (43) calculate the time at which each asset visits nodes as it moves through the network. Within Constraint (43), the term $\omega\tau_{ijk}$ allows assets to slow down to $1/\omega$ of their fastest speed for traversing arc (i, j) to coordinate sequential arrival times at demands. When applied to instances, one may parameterize M to be equal to the longest time required to visit every demand node. Alternatively, it suffices to set $M = \max_{k \in K} \left\{ \sum_{(i,j) \in A} 2\tau_{ijk} \right\}$. Constraint (44) requires $t_{ik} = 0$ if asset k never visits node i.

Constraint (45) bounds $\pi_{i\psi}$-values below by the latest asset of type $k \in K_\psi$ to arrive at node $i \in N^D$. For each asset type, Constraints (46) and (47) collectively impose an upper bound on $\psi_{i\psi}$ to affix the visit to a node by the latest asset of type $k \in K_\psi$. Constraint (48) ensures the visits by asset types do not violate an ordinal sequence, with simultaneous visits allowed. For each demand node, Constraint (49) requires the sequence of visits by asset types to occur within a range of δ units of time.

Finally, constraints (50)-(54) enforce the respective, appropriate restrictions on the decision variables.

As an aside, we note that Problem P has a convenient, alternative use. As formulated, it considers a defender seeking to delay an attacker via route-delaying interdictions at locations $\ell \in L$. Alternatively, consider those locations to be where an attacker may impose additional security measures to protect travel over nearby arcs (i, j), which would allow faster travel due to the enhanced security. For such a case, $\sigma < 0$; the attacker controls the y_ℓ-variables; the objective is a strict minimization problem, and the mixed-integer linear program may be solved directly with a commercial solver. Beyond the scope of this research, one may also modify Problem P as a competitive location and routing model, wherein a defender imposes a set of delaying actions, and an attacker both imposes a set of security measures and routes the assets.

4.1.2 Solution Methodology

It is of merit to analyze the formulation of Problem P to identify an appropriate solution methodology. Bilevel programming problems are $\mathcal{NP}$-hard (Jeroslow, 1985). As a (notably) more complicated variant of a VRP formulation, the lower-level problem embedded within Problem P is also $\mathcal{NP}$-hard (Toth and Vigo, 2002). Thus, the overall complexity of Problem P is σ_2^p-hard (Arora and Barak, 2009).

The $\mathcal{NP}$-hard nature of bilevel programs does not preclude the existence of transformations to yield more readily solvable, equivalent formulations. For a bilevel program with different upper-level and lower-level objective functions, replacing the lower-level problem with its Karush-Kuhn-Tucker optimality conditions (Colson et al., 2007) yields a single-level, nonlinear program that can be solved directly via a commercial solver, subject to certain convexity-related constraint qualifications on the

lower-level problem in Problem P. For a bilevel program like Problem P wherein there is a single objective function in tension (i.e., a zero-sum Stackelberg game), one may take the dual of the lower-level (a.k.a., *inner*) problem (Wood, 1993; Lunday and Sherali, 2012*a*; Lessin et al., 2018) to attain a single-level nonlinear program, again subject to certain convexity conditions on the lower-level problem.

Although the lower-level formulation in Problem P has binary-valued x_{ijk}-variables, the latter of the aforementioned techniques is viable if an integer-valued solution is optimal when these (binary) integer restrictions are relaxed, i.e., if the lower-level formulation's system of constraints exhibits total unimodularity (Nemhauser and Wolsey, 1993). Unfortunately, this requirement does not hold, due to Constraints (42) and (43). These VRP-style constraints encourage decimal-valued solutions in $\boldsymbol{x}$ by splitting flows to attain artificially lower node arrival times $\boldsymbol{t}$ and, in turn, artificially lower demand service times $\boldsymbol{\pi}$ in the objective function. Thus, we must solve the bilevel program rather than a relaxation-induced, single-level transformation.

For a given defender solution $\boldsymbol{y}$, one may solve the ($\mathcal{NP}$-hard) lower-level problem directly via a commercial solver. As such, one may solve Problem P by searching the upper-level feasible region and, for each solution $\boldsymbol{y}$, identify a corresponding, optimal solution to the lower-level problem, i.e., a best-response by the defender. Such a search procedure is akin to searching the first stage in a two-stage, extensive form game tree, where an optimal solution to Problem P corresponds to a subperfect Nash equilibrium.

An exhaustive enumeration of the upper-level feasible region is unwise. For η disrutive actions and $|L|$ possible locations as which they can occur, a defender has $\binom{\eta+|L|-1}{|L|-1}$ feasible solutions. Given the integer-restricted nature of the decision variables $\boldsymbol{y}$, it is possible to search the upper-level feasible region via a branch-and-bound approach (Bard and Falk, 1982), but the exhaustive nature of such a method por-

tends computational tractability issues, given the $\mathcal{NP}$-hard nature of the lower-level problem. Thus, it is of interest to rapidly identify high-quality solutions to the upper-level problem via an effective construction heuristic and seek to improve upon them via an efficient metaheuristic. The remainder of this section proposes a conceptually-motivated construction heuristic, to be applied in isolation or in combination with other metaheuristics, which Section 4.2 will evaluate via computational testing.

We propose three solution methods to search the upper-level feasible region to identify high quality solutions with relative computational efficiency. As Table 8 indicates, these methods respectively consist of a greedy construction heuristic (GCH); GCH followed by simulated annealing (SA); and GCH followed by an enhanced variant of simulated annealing (eSA).

Table 8. Solution Methods Examined

Name	Construction Heuristic	Improvement Metaheuristic
GCH	Greedy	–
SA	Greedy	Simulated Annealing (SA)
eSA	Greedy	Enhanced SA

Whereas the first solution method entails only the identification of a feasible solution using GCH, the second solution method subsequently applies a simulated annealing metaheuristic (Kirkpatrick et al., 1983). We selected simulated annealing as a baseline metaheuristic because GCH provides a single solution upon which to improve. In contrast, population-based metaheuristics (e.g., genetic algorithms (Holland, 1973), particle swarm optimization (Kennedy and Eberhart, 1995), ant colony optimization (Dorigo and Di Caro, 1999)) require more initial candidate solutions, and their generation would require the embrace of deliberately worse construction heuristics or random interdiction strategies, neither of which has conceptual appeal. Other single-solution metaheuristics are certainly available (e.g., Tabu search (Glover, 1986), GRASP (Marques-Silva and Sakallah, 1999)), and a consideration of their

mechanisms informs the third solution method. Subsequent discussion details the components of each solution method and their implementation.

Greedy Construction Heuristic (GCH)

GCH identifies a conceptually effective, feasible solution to Problem P. Whereas greedy heuristics entail no assurance of identifying an optimal solution, a logical series of choices when constructing a feasible solution will often yield a good solution. Given η disruptive actions allowed, GCH identifies a solution by iteratively identifying each subsequent disruptive action location $\ell \in L$ by solving the lower-level problem $(\eta+1)$ times.

Define **Problem P1** as the lower-level problem within Problem P, which seeks to minimize the objective function subject to constraints (39)-(54), given a feasible interdiction strategy $\bar{\boldsymbol{y}}$. Solving P1 identifies an optimal asset routing solution.

Define **Problem P2** as Problem P with the goal of only maximizing the objective function, given both a fixed, partial attacker routing solution $\bar{\boldsymbol{x}}$ and a fixed, partial defender solution $\bar{\boldsymbol{y}}$ where $\sum_{\ell \in L} \bar{y}_\ell = (\eta - 1)$, and with the additional constraint $y_\ell \geq \bar{y}_\ell$, $\forall \ell \in L$. Solving Problem P2 identifies the best location for the η^{th} disruptive action, given $(\eta - 1)$ such actions have been identified and are fixed.

Leveraging these definitions, Algorithm 5 presents GCH. Line 1 initializes the total number of disruptive actions η^* and a null interdiction solution $\bar{\boldsymbol{y}}$. Line 2 identifies an optimal routing solution $\bar{\boldsymbol{x}}$ in the absence of interdiction and updates the current routing solution. Within Lines 3-7, GCH iteratively identifies the location of the disruptive actions. For each such action, Line 4 increments the number of actions η by 1; Line 5 identifies the location of the additional action within the updated interdiction strategy $\bar{\boldsymbol{y}}$; and Line 6 identifies the best asset-routing response $\bar{\boldsymbol{x}}$. Upon termination, Line 8 returns the interdiction strategy $\bar{\boldsymbol{y}}$, feasible to Problem P with

objective function value z^*.

Algorithm 5 Greedy Construction Heuristic

1: Set $\eta^* = \eta$ and $\bar{\boldsymbol{y}} = \mathbf{0}$
2: Solve Problem P1 for $\bar{\boldsymbol{y}}$ to identify $\boldsymbol{x}^*$ and z^*, and let $\bar{\boldsymbol{x}} \leftarrow \boldsymbol{x}^*$
3: **for** *counter* $= 1$ to η^* **do**
4: Set $\eta \leftarrow$ *counter*
5: Solve Problem P2 for $\bar{\boldsymbol{x}}$ to identify $\boldsymbol{y}^*$ and let $\bar{\boldsymbol{y}} \leftarrow \boldsymbol{y}^*$
6: Solve Problem P1 for $\bar{\boldsymbol{y}}$ to identify $\boldsymbol{x}^*$ and z^*, and let $\bar{\boldsymbol{x}} \leftarrow \bar{\boldsymbol{x}}^*$
7: **end for**
8: **return** $\bar{\boldsymbol{y}}$, $\boldsymbol{x}^*$, and z^*

Simulated Annealing Metaheuristic

Originally developed by (Kirkpatrick et al., 1983), simulated annealing is a useful improvement metaheuristic for vehicle routing problems (e.g., Osman, 1993; Van Breedam, 1995; Chiang and Russell, 1996; Vincent et al., 2017). Other interdiction models have explored the use of simulated annealing to achieve near optimal solutions for large scale models (e.g., Janjarassuk and Nakrachata-Amon, 2015; Parsafard and Li, 2021).

Whereas a hill-climbing algorithm or descent method seeks only improving solutions, the simulated annealing metaheuristic allows a move within a feasible region from a current solution to a solution with a worse objective function value. The goal of this allowance is to avoid becoming trapped in a local optimum that would otherwise preclude a broader search of the feasible region, thereby improving the likelihood of identifying a global optimum. Such a wariness of converging to a local-but-not-global optimum is warranted for nonconvex optimization problems, including mixed-integer linear programming problems like Problem P.

Each iteration of a conventional implementation of SA functions as follows. Given a feasible solution and a defined *neighborhood* of solutions, identify and evaluate a candidate solution in the neighborhood. If the candidate solution is feasible and has an improved objective function value, accept it as the new solution with certainty;

otherwise, accept it as the new solution with some probability p. As SA proceeds, the metaheuristic deliberately reduces p, typically fast initially and then slower. Much like the malleability of a metal that is cooled via an annealing process, the willingness of SA to adopt a worse candidate solution reduces over time (i.e., iterations), eventually hardening (i.e., converging) to the behavior of a hill-climbing algorithm or descent method. SA typically terminates after a fixed, user-defined number of iterations or when a temperature parameter T used to calculate p decreases below a predetermined value T^* (Bard, 2013). Affecting its performance notably, an SA implementation has two important characteristics: (i) its definition of a neighborhood and (ii) a probability function with associated parameters to define the probability p.

SA Neighborhood Definition Given a feasible interdiction strategy $\bar{\boldsymbol{y}}$ for Problem P, we define the neighborhood to be

$$\mathcal{Y}(\bar{\boldsymbol{y}}) \equiv \left\{ \boldsymbol{y} : \sum_{\ell \in L} y_\ell = \eta; y_\ell \in \mathbb{Z}_+, \forall\ \ell \in L; \sum_{\ell \in L} |\bar{y}_\ell - y_\ell| = 2 \right\}$$

In practice, one can identify a candidate solution $\boldsymbol{y}' \in \mathcal{Y}(\bar{\boldsymbol{y}})$ by selecting a single location $\tilde{\ell}$ in the current solution having $y_{\tilde{\ell}} \geq 1$ and moving a disruptive action from that location to any other location $\ell' \in L \setminus \{\tilde{\ell}\}$. Our implementation of SA identifies a candidate solution $\boldsymbol{y}'$ in this manner, selecting the location $\tilde{\ell}$ from a discrete uniform distribution over all disruptive action locations and the new location for a disruptive action via a discrete uniform distribution over $\ell \in L \setminus \{\tilde{\ell}\}$.

Solving P1 for the candidate solution $\boldsymbol{y}'$ yields an optimal objective function value z'. Denoting the current solution's optimal objective function value for P1 is $\bar{z}$, the relative (%) objective function value increase is $\Delta = (z' - \bar{z})/\bar{z}$.

SA Probability Function The probability function defined in Equation (55) computes the likelihood p with which SA accepts a candidate solution $\boldsymbol{y}'$ within a given iteration, as a function of both Δ and a declining temperature parameter, T. As indicated by the conditions on Δ, our SA implementation accepts any candidate solution that is *not worse* than the current solution with certainty, and it accepts a worse solution with probability p, declining exponentially on (Δ/T) for negative values of Δ.

$$p = \begin{cases} 1 & \Delta \geq 0 \\ e^{(\Delta/T)} & \Delta < 0 \end{cases} \tag{55}$$

Of note, T is not a fixed parameter; initialized with a temperature $T = T_0$, it decreases with each iteration to affect the annealing process for any fixed $\Delta < 0$. Many functional forms (e.g., linear, geometric) are available to update the temperature T. Based upon initial computational testing for instances of Problem P, we adopted the temperature update function given by Equation (56) for a user-determined parameter $\beta > 0$.

$$T \leftarrow \frac{T}{1 + \beta T} \tag{56}$$

Defining the temperature threshold and maximum iteration count used for alternative SA termination criteria as T^* and $iter_{max}$, respectively, Algorithm 6 presents our implementation of the SA metaheuristic to solve an instance of Problem P, returning a best identified interdiction strategy $\boldsymbol{y}^*$, the corresponding best routing response $\boldsymbol{x}^*$, and the resulting objective function value z^*.

Within Algorithm 6, Line 1 recognizes the initial feasible solution, initial temperature parameter, and the two alternative termination criteria. Line 2 identifies the best routing response for the interdiction strategy and the corresponding objective function value. Given an initial solution identified via GCH, Line 2 is not necessary, but we retain it to represent SA for alternative means to identify an initial feasible

Algorithm 6 SA Implementation

1: Given $\bar{\boldsymbol{y}}$ feasible to Problem P1, T_0, T^*, and $iter_{max}$
2: Solve P1 for $\bar{\boldsymbol{y}}$ to identify $\bar{\boldsymbol{x}}$ and $\bar{z}$
3: Let $\boldsymbol{y}^* \leftarrow \bar{\boldsymbol{y}}$, $\boldsymbol{x}^* \leftarrow \bar{\boldsymbol{x}}$, $z^* \leftarrow \bar{z}$, and $T \leftarrow T_0$
4: **for** $iter = 1$ to $iter_{max}$ **do**
5: Update T via Equation (56)
6: Identify a candidate solution $\boldsymbol{y}' \in \mathcal{Y}(\bar{\boldsymbol{y}})$
7: Solve P1 for $\boldsymbol{y}'$ to identify $\boldsymbol{x}'$ and z', and compute Δ
8: With probability p via Equation (55), let $\bar{\boldsymbol{y}} \leftarrow \boldsymbol{y}'$, $\bar{\boldsymbol{x}} \leftarrow \boldsymbol{x}'$, and $\bar{z} \leftarrow z'$
9: **if** $\bar{z} > z^*$ **then**
10: Let $\boldsymbol{y}^* \leftarrow \bar{\boldsymbol{y}}$, $\boldsymbol{x}^* \leftarrow \bar{\boldsymbol{x}}$, $z^* \leftarrow \bar{z}$
11: **end if**
12: **if** $T < T^*$ **then**
13: **break**
14: **end if**
15: **end for**
16: **return** $\boldsymbol{y}^*$, $\boldsymbol{x}^*$, and z^*

solution (e.g., a randomly generated interdiction strategy). Line 3 initializes the incumbent solutions and cooling temperature. Within Lines 4-15, SA identifies and evaluates at most $iter_{max}$ candidate solutions. Line 5 updates the cooling temperature. Line 6 identifies a candidate solution, and Line 7 evaluates it. Line 8 determines whether to accept the candidate solution as the current solution, and Lines 9-11 update the incumbent solution if appropriate. An iteration concludes by comparing if T is lower than the threshold T^*, and, if so, Lines 12-14 terminate the for loop. Otherwise, iterations continue until $iter = iter_{max}$, after which the procedure concludes in Line 16 with the best identified interdiction strategy, routing response, and objective function value.

Enhanced Simulated Annealing Metaheuristic

Within Algorithm 6, Line 6 incurs a notable computational expense. As previously discussed, Problem P1 is $\mathcal{NP}$-hard. The Enhanced Simulated Annealing (eSA) metaheuristic seeks to reduce the number of times it solves Problem P1 without a

productive exploration of the upper-level feasible region. More specifically, SA exhibits two inefficiencies, as implemented in Algorithm 6. First, a candidate solution $\boldsymbol{y}'$ may have been previously explored. Second, the generation of a candidate solution $\boldsymbol{y}'$ from a current solution $\bar{\boldsymbol{y}}$ may remove an effective disruptive action while adding an ineffective action elsewhere. The eSA metaheuristic addresses these computational inefficiencies via two mechanisms: (i) a tabu list and (ii) an alternative method to generate candidate solutions.

Borrowing from the Tabu Search metaheuristic (Glover, 1986), eSA maintains a bounded tabu list of the most recently examined interdiction solutions. If a candidate solution $\boldsymbol{y}'$ is on the list, eSA identifies an alternative candidate solution to evaluate. Other interdiction research (e.g., Michalopoulos et al., 2015; Aksen and Aras, 2013) has directly applied Tabu Search, so there is precedent for a tabu list for reducing computational effort when solving a bilevel program.

With regard to generating a candidate solution in Line 6 of Algorithm 6, eSA discards the randomized mechanisms for moving a disruptive action to a new location, as discussed in Section 4.1.2. It instead embraces a conceptually greedy approach, seeking to move a (perceived) least effective disruptive action to a location where it would be most effective. Given a solution $\bar{\boldsymbol{y}}$, only $\tilde{\ell}$ and ℓ' are necessary to define the candidate solution $\boldsymbol{y}'$. Algorithm 7 defines the eSA process to identify $(\tilde{\ell}, \ell')$ and, in turn, $\boldsymbol{y}'$.

Within Algorithm 7, we seek to identify the pair of locations $(\tilde{\ell}, \ell')$ that preemptively and respectively remove a disruptive action from the location affecting arcs least (currently) travelled by assets and move it to a location that will affect arcs most (currently) travelled by assets, such that the resulting solution $\boldsymbol{y}'$ is not on the tabu list. Line 1 recognizes the given information, and Lines 2 and 3 define the sorted lists R1 and $\text{R2}_{\tilde{\ell}}$. If there is a tie when sorting any list, the respective ordering of

Algorithm 7 eSA Identification of $\boldsymbol{y}'$

1: Given a tabu list $\mathcal{T}$, an interdiction strategy $\bar{\boldsymbol{y}}$, and $\boldsymbol{x}^*$ as the corresponding solution to Problem P1
2: Define R1 as a list of current disruptive action locations $\tilde{\ell}$, sorted in ascending order on $\sum_{(i,j)\in A}\sum_{k\in K}\alpha_{ij\tilde{\ell}}x^*_{ijk}$
3: Define R2$_{\tilde{\ell}}$ as a list of potential disruptive action locations $\ell' \in L \setminus \{\tilde{\ell}\}$, sorted in descending order on *strictly positive* values of $\sum_{(i,j)\in A}\sum_{k\in K}\alpha_{ij\ell'}x^*_{ijk}$
4: **for** $\tilde{\ell} \in$ R1 **do**
5: **for** $\ell' \in$ R2$_{\tilde{\ell}}$ **do**
6: **if** $\boldsymbol{y}' \notin \mathcal{T}$ **then**
7: **return** $\boldsymbol{y}'$
8: **end if**
9: **end for**
10: **end for**

indices having the same score is random. Lines 4-10 identify $\boldsymbol{y}'$, wherein Lines 4 and 5 assume the iteration of elements in each list in the respective sorted orders. If Algorithm 7 terminates without identifying a $\boldsymbol{y}' \notin \mathcal{T}$, the eSA metaheuristic terminates.

Algorithm 7's identification of $(\tilde{\ell}, \ell')$ merits further discussion. Selecting $\tilde{\ell}$ via R1 is an imperfect greedy approach. From an ideal perspective, it will move an ineffective disruptive action in the current interdiction solution. Alternatively, it may move a disruptive action that was so effective that the lower-level decision-maker avoided having agents traverse any nearby arcs. The possibility that $\tilde{\ell}$ is poorly selected may provide an equivalent or better asset routing when a disruptive action is moved to ℓ', resulting in $\Delta < 0$. As such, the sound conceptual motivation for Algorithm 7 implemented with eSA does not improve upon SA with certainty. Computational testing is necessary to assess their relative performances, as Section 4.2 examines.

4.2 Testing, Results, and Analysis

For a network $G(N, A)$, we consider a regular hexagonal tessellation of a planar region, a discretization technique embraced in the literature for routing assets in a

region not restricted to a road network, such as airspace (e.g., Yousefi and Donohue, 2004; Lunday et al., 2012; Lessin et al., 2018). Additionally, a hexagonal grid provides benefit of well defined areas for disruptive actions, i.e., $\ell \in L$ as center of a hexagon, wherein a disruptive action (i.e., where $y_\ell \geq 1$) slows travel across all arcs bordering the hexagon.

4.2.1 Illustrative Example

For the illustrative example, Figure 18 presents the hexagonal mesh with 16 regular hexagons (i.e., $|L| = 16$). Depicted on the network are $K = \{1, 2\}$ assets and $\Psi = \{1, 2\}$ asset types, with $K_1 = \{1\}$ and $K_2 = \{2\}$ having respective origins indicated by the blue circle at Node 31 and purple diamond at Node 3. Red stars indicate the set of demand points $N^D = \{12, 26, 34\}$. Asset Types 1 and 2 have different minimum travel times of $(\tau_{ij1}, \tau_{ij2}) = (1, 0.8)$, $\forall\ (i, j) \in A$, and the maximum range of time between when assets of Type 1 and 2 must respectively service a demand is $\delta = 5$.

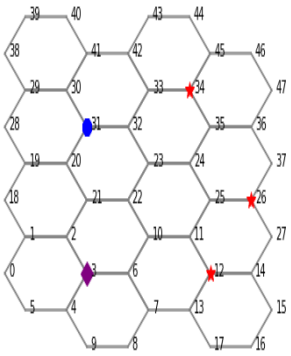
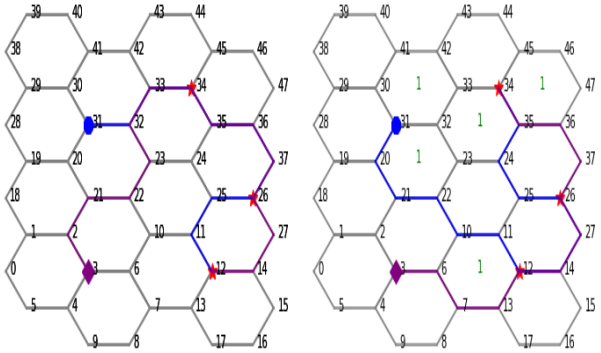


Figure 18. Illustrative Example: Hexagonal Mesh, Initial Asset Locations, and Demand Node Locations

Figure 19a illustrates an optimal routing of the assets to service demands in the absence of delaying actions implemented by the defender (i.e., $\eta = 0$), corresponding to an objective function value of 25.6. Figure 19b presents the delaying actions

found via GCH with $\eta = 5$, and the corresponding optimal asset routing solution for the lower-level problem. The delaying actions increase travel time on the affected arcs by $\sigma = 1.5$ time units. Figure 19b depicts numerals (e.g., 1) in the center of hexagons where $y_\ell = 1$, indicating the number of delaying actions at the center of selected hexagons, in turn increasing travel times on the bordering the arcs. With the $\eta = 5$ disruptions depicted, the optimal asset routing solution has a minimal objective function value of 55.



(a) Routing Solution No Delays **(b) Optimal Routing $\eta = 5$**

Figure 19. Example Model Execution

Accompanying the visual depictions in Figures 19a and 19b, Tables 9 and 10 report the π_i-values resulting from the arrival times of asset types $\psi = 1$ (i.e., asset $k = 1$) and $\psi = 2$ (i.e., asset $k = 2$) at each of the demand nodes. Examining the rightmost columns, the GCH-identified delaying actions increase the cumulative service time by 29.4, resulting from delaying the servicing of Nodes 12, 26, and 34 respectively by 2.3, 7.7, and 24.4 time units. Also compelling, the disruptive actions change the optimal routes of the assets, reverse the order in which the demands are serviced, and compel simultaneous servicing of demands by both assets.

As an aside, there do exist alternative optima for the asset routing. The two paths traversing from Node 26 to Node 35 (i.e., $26 - 25 - 24 - 35$ and $26 - 37 - 36 - 35$) in Figure 19b are equivalent. Such a characteristic is more likely to exist in optimal

Table 9. Demand Service Times with No Delays

Demand Node	Asset Type $\psi = 1$	$\psi = 2$
12	10.0	11.2
26	7.0	8.8
34	3.0	5.6

Table 10. Demand Service Times with GCH-identified $\eta = 5$ Disruptive Actions

Demand Node	Asset Type $\psi = 1$	$\psi = 2$
12	13.5	13.5
26	16.5	16.5
34	25.0	25.0

asset routing solutions for instances having common arc traversal times and a graph induced via a regular tessellation of a region, which collectively contribute to problem symmetry. Given this observation, specialized techniques such as orbital branching (Ostrowski et al., 2011) may reduce the required computational effort to solve the lower-level problem, although the exploration of such techniques is beyond the scope of this study.

4.2.2 Parameter Exploration

To evaluate the potential of both SA and eSA to improve upon GCH-identified solution, testing considered alternative numbers of delaying strategies (η), initial temperature values (T_0), and the cooling rate parameter (β) used in the temperature update function, as discussed in Section 4.1.2. Table 11 displays the parameters values explored herein.

Table 11. Values to Explore

Factor	Values Explored
Number Delaying Actions η	$3, 4, 5, 6$
Initial Temperature T_0	$1, 5$
Cooling Rate Parameter β	$0.01, 0.1$

For the baseline instance depicted in Figure 19a, testing examined both SA and eSA with combination of the parameters in Table 11, initializing them with a GCH-identified solution having the same η-value and using a common random seed. The effect of high and low values of both T_0 and β explored the respective impacts of the

temperature update function and the probability of accepting a candidate solution that will not yield an immediate improvement. Both SA and eSA terminated after 45 iterations or when the annealing temperature dropped below 0.05, whichever came first. All testing was performed on an Intel(R) Core(TM) i7-10875H CPU @2.30GHz with 128 GB of RAM on a 64-bit operating system, and using Python (Version 3.9.7) with the GurobyPi package to invoke the commercial solver Gurobi (Version 9.5.1). When solving the lower-level problems using Gurobi, alternative termination criteria were 1800 seconds (i.e., 30 minutes) of computational effort and an identified 0.0005% relative optimality gap.

Figure 20 depicts four temperature update functions over the respective (T_0, β)-combinations over 45 iterations. It depicts faster initial decreases in temperature for larger values of β or T_0. Of note, the combination of $(T_0, \beta) = (5, 0.01)$ never exhibits a temperature below 1, even after 45 iterations.

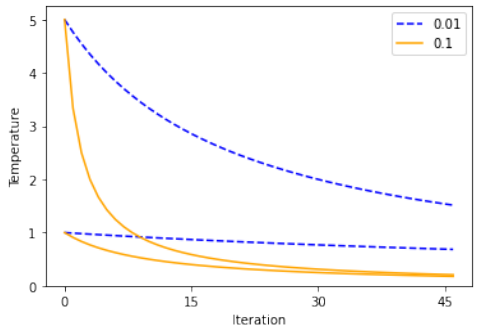


Figure 20. Annealing Temperature as a function of T_0 and β

Table 12. Probability of Accepting Worse Candidate Solution (%) via Equation 55

		Δ			
		-0.05	-0.1	-0.2	-0.4
	4	99.8	97.5	95.1	90.5
	2	97.5	95.1	90.5	81.9
T	1	95.1	90.5	81.9	67.0
	0.5	90.5	81.9	67.0	44.9
	0.25	81.9	67.0	44.3	20.2

If a candidate solution improves the currently accepted solution, either SA or eSA algorithm will accept it with certainty; otherwise the probability of acceptance is determined by current temperature, T, and the change between the current solution and candidate solution, Δ. Table 12 presents the probability (%) of accepting a worse candidate solution over a sample of (T, Δ)-value combinations. When T is greater than 1, the probability of accepting the candidate solution is greater then 0.5, even

if the solution is notably worse. When $T < 1$, acceptance of the candidate solution depends on a lower magnitude for Δ. This leads to an understanding that a high T_0-value paired with a low β-value will more likely accept worsening candidate solutions, regardless of how much worse they are. Conversely, a low T_0 paired with a high β-value are less likely to accept a worse candidate solution, even if the magnitude of its Δ-value is small.

4.2.3 Comparative Testing of Solution Methods

Table 13 reports the objective function value attained via GCH for each of the η-values explored. Because the GCH algorithm iteratively adds a delaying strategy to the solution, the objective function value strictly increases with η and does so at a reasonably steady rate. The remainder of this section explores the relative performances of SA and eSA. The GCH-identified y_ℓ-values initialize both the SA and eSA algorithms.

Table 13. Objective Function Values for Solutions Identified via the Greedy Construction Heuristic

η	Objective
3	42.7
4	47.0
5	55.0
6	62.2

Table 14 presents the best objective function value found after 45 iterations of SA and eSA for each combination η, T_0, and β from Table 11. SA identified an improved solution over GCH for 12 of the 16 instances. It failed to do so only when $T_0 = 1$, three times of which occurred with $\beta = 0.01$, indicating the relatively poor performance of that parametric combination for SA. This outcome is likely due to the lower probability of accepting a worse candidate solution (i.e., getting stuck in a local-but-not-global optimal solution). In contrast, eSA improved upon the GCH

solution at each η-value and for every combination of (T_0, β)-parameters.

Table 14. Best Objective Function Value after 45 Iterations for the SA and eSA Algorithms

		$\eta = 3$		$\eta = 4$		$\eta = 5$		$\eta = 6$	
T_0	β	SA	eSA	SA	eSA	SA	eSA	SA	eSA
1	0.01	40.8	48.1	49.7	54.1	55.0	61.6	61.0	69.4
	0.10	48.1	48.1	49.7	54.1	62.5	61.6	61.0	69.4
5	0.01	46.8	48.1	49.7	55.0	55.0	61.6	69.0	70.6
	0.10	48.1	48.1	49.7	54.1	55.0	61.6	69.0	67.6

Whereas a higher initial temperature portends better outcomes, the effect of β on the best identified objective function value is more nuanced. Higher β-values performed the same or better when $T_0 = 1$, and lower β-values more often did better when $T_0 = 5$, albeit not universally (i.e., SA did better with $(T_0, \beta) = (5, 0.10)$ than $(5, 0.01)$ with $\eta = 3$). Broader conclusions from the results in Table 14 are elusive, and Section 4.2.4 reports the results of additional testing specific to the effect of alternative β-values.

Overall, these results compel an examination of whether the annealing aspect of the SA and eSA algorithms is effective. That is, for these instances, would SA or eSA perform better by either *never* accepting a worse candidate solution (i.e., $p = 0$) or *always* accepting it (i.e., $p = 1$)? Table 15 reports the objective function value of solution respectively identified by SA and eSA for $p = 0, 1$ and $\eta = 3, 4, 5, 6$.

Table 15. Best Objective Function Value after 45 Iterations for the SA and eSA Algorithms in the Absence of Annealing, i.e., with Fixed Probability p of Accepting a Worse Candidate Solution

	$\eta = 3$		$\eta = 4$		$\eta = 5$		$\eta = 6$	
p	SA	eSA	SA	eSA	SA	eSA	SA	eSA
0	46.6	46.6	54.3	55.6	60.0	61.8	66.7	66.7
1	46.8	48.1	49.7	55.0	55.0	61.6	69.0	70.6

Within the results in Table 15, eSA performed as well or better than SA for each instance. However, neither never ($p = 0$) nor always ($p = 1$) accepting a worse

candidate solution was a universally beneficial modification to either SA or eSA. Comparing the results with Table 14, eSA with $p = 0$ identified the best solution for $\eta = 4$, and eSA with $p = 1$ did so for $\eta = 6$. These results reinforce the merit of exploring alternative annealing schemes and parametric combinations.

For the testing reported in Tables 14 and 15, Table 16 presents the relative performance of eSA and SA regarding the objective function value and the required algorithmic runtime. More specifically, it tabulates the percentage of instances for which eSA performed as well as or better than SA. The first row aggregates the results over all combinations of (T_0, β)-values and p-values, and over all η-values. The second and third rows partition the results by $\eta = 3, 4$ and $\eta = 5, 6$, respectively.

Table 16. Instances (%) for which eSA performed as well or better than SA

η-values	Objective Function Value	Runtime
3,4,5,6	91.7	66.7
3,4	100.0	75.0
5,6	83.3	58.3

The eSA algorithm found equivalent or better solutions than SA for 91.7% of the instances of different η-values and parametric combinations. There were two instances where SA found a strictly better solution, at $\eta = 5$ and 6, both when $\beta = 0.1$. This result improved to 100% when restricted to smaller η-values. The eSA had a faster runtime for more than half of the instances, which improved to 75% when considering only the lower η-values.

Additionally, testing examined the relative performance of eSA and SA over five different starting seeds for pseudorandom number generation. Not reported in detail here, testing found that eSA always found the same or better solutions than SA, and it found them more consistently; eSA found the same best solution across all random seeds explored, whereas SA identified a distinctly different best solution for each random seed.

4.2.4 Selected Excursional Analyses

Additional testing examined the effects of the different values of β, as well as the quality of GCH-identified solutions and instance tractability for routing multiple assets of each type and when routing a third asset type.

Sensitivity Analysis for β-values

Testing results in the Section 4.2.3 indicated the merit of having higher T_0-values to more likely accept worse candidate solutions. Affixing $T_0 = 5$, testing investigated the effects of β-values of 0.050 and 0.075. Figure 21 shows the corresponding temperature update functions over 45 iterations. The figure demonstrates that the $T = 1$ threshold, a value where rejection of a worse candidate solution occurs more frequently, occurs at approximately 30 and 15 iterations respectively for $\beta = 0.050$ and 0.075.

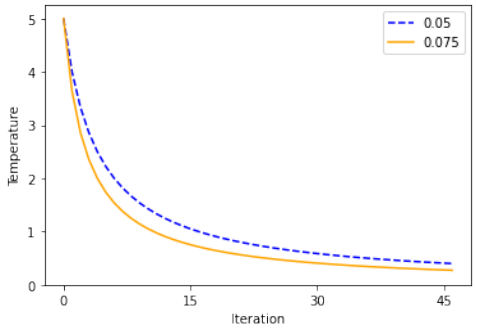


Figure 21. Annealing Temperature for $T_0 = 5$ as a function of β

Table 17 presents the best objective function value identified by the respective SA and eSA algorithms after 45 iterations for the β-values explored with $T_0 = 5$. In all instances except one (i.e., $\beta = 0.075$ and $\eta = 6$), eSA found a solution that was equivalent to or better what SA identified.

Based on the results in Table 17 and previous testing, we recommend the use of eSA with a higher T_0-value and a lower β-value. However, even these results exhibit

Table 17. Best Objective Function Value after 45 Iterations for the SA and eSA Algorithms with $T_0 = 5$ and $\beta = 0.075$

		$\eta = 3$		$\eta = 4$		$\eta = 5$		$\eta = 6$	
T_0	β	SA	eSA	SA	eSA	SA	eSA	SA	eSA
5	0.050	40.8	48.1	49.7	54.1	55.0	61.6	69.0	70.6
	0.075	48.1	48.1	49.7	54.1	55.0	61.6	69.0	67.6

nuance. For lower values of η (i.e., 3 or 4), $\beta = 0.075$ was able to improve the GCH objective-value reliably. Similarly, higher η-values of 5 or 6 performed well when $\beta = 0.05$.

Additional Assets for each Asset Type

Additional testing explored a modified instance having $K = \{1, 2, 3, 4\}$ assets and $\Psi = \{1, 2\}$ asset types, wherein $K_1 = \{1, 2\}$ and $K_2 = \{3, 4\}$. Figure 22 depicts the respective origins indicated by the blue circles at Nodes 31 and 41 and purple diamonds at Nodes 3 and 9. All other aspects of the instance are unchanged from Section 4.2.1. We also restrict our attention to only eSA vis-á-vis GCH, based upon the relatively poorer performance of SA.

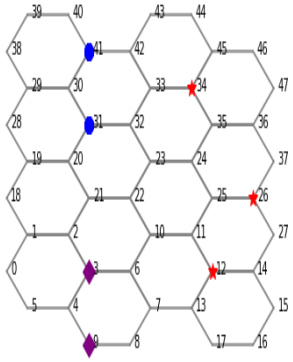
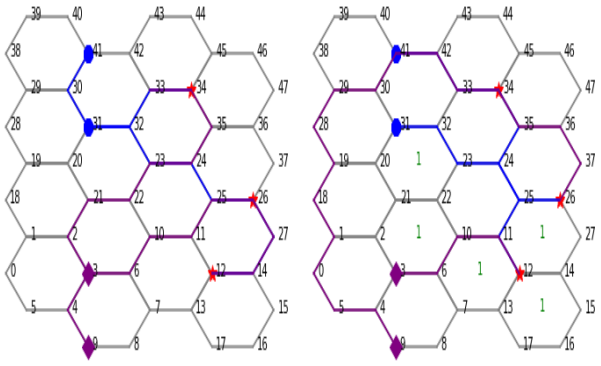


Figure 22. Modified Illustrative Example: Hexagonal Mesh, Initial Asset Locations, and Demand Node Locations

Figures 23a and 23b respectively present the asset routing solutions for no disruptive actions ($\eta = 0$) and the GCH-identified solution when $\eta = 5$. The asset routing

is visually different. In Figure 23a, assets of type $\psi = 1$ travel directly to demand nodes, whereas assets of type $\psi = 2$ initially meander to allow for sequential servicing of the demands. Moreover, the assets primarily traverse arcs in the middle of the network. In contrast, within Figure 22 where $\eta = 5$, most of the disruptive actions slow the travel of assets of type $\psi = 2$, which no longer meander, although more assets traverse arcs on the boundary of the graph to avoid slower travel.



(a) Routing Solution No Delays **(b) Optimal Routing $\eta = 5$**

Figure 23. Example Model Execution Additional Assets

For each η-value examined, subsequent pairs of columns within Table 18 report for GCH and eSA (after 45 iterations with $(T_0, \beta) = (5, 0.05)$) the best objective function value identified ($\bar{z}$) and the required algorithmic runtime (seconds).

Table 18. Best objective function values identified via GCH and eSA after 45 iterations with $(T_0, \beta) = (5, 0.05)$ for the instance depicted in Figure 22 with $|K_1| = |K_2| = 2$, for increasing η-values

	GCH		eSA	
η	$\bar{z}$	time (sec)	$\bar{z}$	time (sec)
3	34.2	7208.3	35.9	81075.0
4	34.2	9010.6	41.4	81081.0
5	36.5	10812.5	44.9	81076.0
6	44.2	12615.0	50.7	81061.9

GCH managed smaller improvements as η increased, with no increase between $\eta = 3$ and 4, indicating alternative optimal routing solutions at $\eta = 3$. The best

objective function found after 45 iterations of eSA consistently improved upon GCH results. The relative (%) improvement in eSA solution quality over GCH solution quality compared to the relative (%) increase in runtime merits discussion. Compared to GCH, eSA attained a minimum, average, and maximum increase in the objective function value by 5.0%, 15.9%, and 23.0%.

This improvement required a six to eleven-fold increase in runtime. For challenging problem instances, this ratio is foreseeable. The additional asset added to each asset type complicated the solver's ability to reach optimality for Problem P1 instances, even with a 30-minute runtime. Given the time limits, the maximum time to apply GCH for η disruptive actions is $(1+\eta)1800$ seconds, plus the relatively small amount of time to construct formulations, retrieve solutions, and manage stored information. This equates to $\sim$ 7200, $\sim$ 9000, $\sim$ 10800, and $\sim$ 12600 seconds. Evident from the results in Table 18, Gurobi is terminating due to time limitations for each GCH instance of Problem P1.

Note that the eSA runtime is the runtime *after* it is initialized with the GCH-identified solution. Given 45 iterations with an 1800 time limit to solve each lower-level problem, the maximum eSA time is $\sim$81,000 seconds, plus the time to search the upper-level feasible region, construct formulations, retrieve solutions, and manage stored information. Thus, for eSA instances, Gurobi is again terminating due to time limits. More specifically, each eSA implementation required 22.5 hours of runtime, whereas GCH required 2, 2.5, 3, and 3.5 hours of runtime for $\eta = 3, 4, 5, 6$.

Whereas results presented in Section 4.2.3 exhibited an average relative optimality gap $< 0.01\%$ for lower-level problem instances, these results averaged 73.9%. Of note, such results do not indicate the solutions to the lower-level problems are necessarily suboptimal; they demonstrate the solutions to be no worse than indicated, but they may be better. In general, they illustrate the $\mathcal{NP}$-hard nature of the lower-level

problem, and Gurobi must be considered as a heuristic under the runtime limitations.

In general, eSA has merit if the planning time to identify disruptive actions allows enough time to use it. However, it is worth noting that eSA found the best improved objective within the first 15 iterations (∼7.5 hours) for $\eta = 3$ and 6 and within the first 30 iterations (∼15 hours) for $\eta = 4$ and 5. Moreover, eSA still attained improvements of 17.8% and 19.5% for $\eta = 4$ and 5 after 15 iterations, so a decision to embrace eSA is not binary. We recommend its use to improve upon GCH solutions for any time that can be afforded to identify a high qualitiy network disruption strategy.

Additional Asset Type $|\Psi| = 3$

An additional experiment examined a test instance to route one each of $|\Psi| = 3$ different types of assets. Figure 24 presents the instance, for which the only difference from Figure 18 is an additional asset $K_3 = \{3\}$ starting at Node 18, depicted via a green cross. This asset has a minimum travel time $\tau_{ij3} = 0.9, \forall (i,j) \in A$, a fastest speed equal to the average of τ_{ij1} and τ_{ij2}.

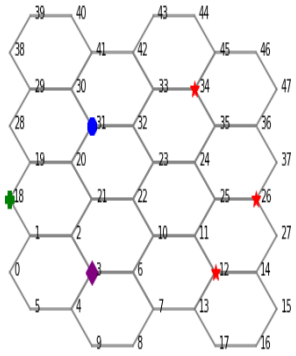
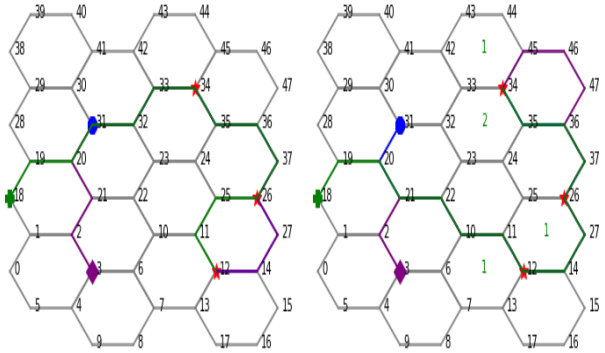


Figure 24. Illustrative Example: 3 Asset Types

Figures 25a and 25b respectively present the routing solutions for no disruptive actions ($\eta = 0$) and the GCH-identified solution when $\eta = 5$. Of interest is that both solutions present a routing solution wherein, after each asset reaches a common node (Node 31 in Figure 25a and Node 21 for in Figure 25b), they traverse the

same path to the targets, sequenced in ascending order by asset type. A noteworthy difference between the routing solutions is evident with no disruptive actions ($\eta = 0$) in Figure 25a; the solution routes assets to service demands at Nodes 34, 26, and 12, in sequence. By comparison, the routing solution in Figure 25b (with $\eta = 5$) routes assets to service the demands in reverse order. In the latter solution, the disruptive actions also cause the assets to route on more arcs, including some affected by disruptive actions.



(a) Routing Solution No Delays **(b) Optimal Routing $\eta = 5$**

Figure 25. Example Model Execution $|\Psi| = 3$

Table 19 reports respective objective function values and runtimes for GCH and eSA, again after 45 iterations with $(T_0, \beta) = (5, 0.05)$). Improvement remains consistent over $\eta = 3, 4, 5$, with the largest increase in the objective function value occurring when incrementing η from 5 to 6. Thus, the marginal effect of an additional disruptive action does not follow a predictable trend.

The trade-off between eSA and GCH vis-á-vis solution quality and algorithmic runtime merits discussion. With respect to solution quality, eSA improved upon GCH solutions by a minimum, average, and maximum of 11.7%, 17.3%, and 20.8%, respectively. However, the marginal improvement over GCH strictly declined with an increase in η, indicating the challenge of finding improving solutions for instances with greater complexity.

Table 19. Best objective function values identified via GCH and eSA after 45 iterations with $(T_0, \beta) = (5, 0.05)$ for the instance depicted in Figure 24 with $|\Psi| = 3$ asset types, for increasing η-values

	GCH		eSA	
η	$\bar{z}$	time (sec)	$\bar{z}$	time (sec)
3	43.2	4414.1	52.2	64861.6
4	47.4	7245.3	56.7	61577.5
5	53.8	7768.9	62.7	55811.9
6	64.2	9569.7	71.7	55087.5

Interestingly, this marginal decrease with increasing η-values was accompanied by an opposite trend in runtimes. Comparing the runtimes in Table 19 with Table 18, it is evident that Gurobi is not always terminating due to the 1800-second time limit on Problem P1. Although eSA required from 6 to 15 times as much time as GCH to run for 45 iterations, the lower relative increases compared to GCH occurred for larger η-values. Overall, the solutions to the lower-level problems for eSA applied to all η-values were assuredly closer to optimal than those found in Section 4.2.4. The relative optimally gap averaged 7.3% when finding best response routing solutions to identified disruption strategies.

In general, eSA did find improvement within the first 15 iterations; for only one third of the algorithmic runtimes reported in Table 19, eSA improved over GCH by $10.4\%, 14.6\%, 16.5\%$, and 3.3% respectively for $\eta = 3, 4, 5$, and 6. Thus, eSA retains merit for use, even when the larger runtimes are not available to explore the upper level feasible region more extensively.

4.3 Conclusions and Recommendations

For a subclass of vehicle routing in which multiple asset types are required to service demands in predetermined sequential order over a contested network, this research formulated the problem as a bilevel program, wherein the lower-level problem routes different types of asset types to satisfy demands while minimizing the

cumulative service times. Simultaneously, the upper-level problem seeks to identify a strategy that imposes a bounded number of disruptive actions that slow down agent travel on subsets of the network, seeking to maximize the cumulative service time of demands.

This research set forth and evaluated three solution methods: a greedy construction heuristic, a simulated annealing metaheuristic, and an enhanced simulated annealing that leverages a priority-based candidate solution identification and a tabu list of previously considered solutions. Both simulated annealing-based methods improved upon GCH solutions over a range of problem and algorithmic parameters, with the latter technique doing so more consistently.

Additional testing showed that a larger number of assets of each type was computationally cumbersome, whereas an increased number of asset types was less so. In both excursions, eSA improved the initial solution GCH notably, and it did so within the first 15 iterations (7.5 hours) of runtime.

A natural extension of this work would focus on improving the computational effort required to solve the lower-level problem for a fixed disruption strategy. Although outside the scope of this research, decomposition methods merit study. Within Problem P1, the routing of agents is almost separable, except for the sequencing of assets to service each demand within a bounded time window. Such a structure lends itself to subproblems and a restricted master problem, giving promise to improved tractability for these and larger instances, in turn reinforcing the general solution procedure for the bilevel program.

V. Conclusions and Future Recommendations

Adversarial strategy and technology might have advanced to match and surpass the United States military in some areas. To continue to be competitive and support its allies, the United States military strategy has required rethinking previous methods of warfare and repurposing existing equipment and vehicles in new ways. Where Mosaic Warfare seeks to keep the competitive edge while utilizing current inventory, the complex mechanics of routing and communication maintainability require study and research. This task involves routing asset types with specific roles and capabilities to targets, ensuring prerequisites or arrival requirements are met. Regarding civil application, routing natural disaster supply relief and reconstruction efforts involve similar routing procedures with affected regions and towns. This research explored three problems relating to complex multi-asset asset routing, developed contributing solution methodologies, and explored results based on designed test instance data.

First, this research developed and evaluated a Hierarchical Asset Tiling and Routing Heuristic (HATRH) to implement Mosaic Warfare for an enterprise of aerial assets comprised of airborne sensors, command, and control aircraft, and strike aircraft seeking to move towards and destroy a set of stationary targets. The analysis explored the inefficiencies of routing and munitions required for dynamic asset movement in terms of individual and group asset agency. HATRH explored a practical implementation of a decentralized, regional command-and-control when centralized control may not be feasible. Assets performed algorithmic movements to visit all stationary targets maintaining necessary proximal relationship distance.

HATRH showed across multiple scenario types that inefficiencies increased as operational congestion increased. The user-defined parameter to determine the frequency of asset grouping showed little-to-no effect on performance efficiency. In contrast, a similar user-defined parameter preserving movement allowed to individual assets rel-

ative to group movement did affect performance relative to scenario type, resulting in a recommended default range of this HATRH parameter given asset-target placement for a given scenario.

Next, this research developed and explored a mixed-integer linear math programming formulation for the collaborative vehicle routing problem with proximity service (CoVRP-PS). This model required multiple asset types to service demand collaboratively, with a subset of asset types being allowed proximal deliveries. Adjustable time windows enforced the near-simultaneous arrival to model the collaborative behavior of assets. Instances developed for initial testing of CoVRP-PS are intended to significantly challenge the identification of high-quality, feasible solutions by a leading commercial solver, highlighting problem parameters. Testing also informed the development of two permutations of a model decomposition heuristic and two preprocessing techniques designed to improve performance in identifying a higher quality and larger quantity of solutions.

A designed experiment compared testing of the nine resulting solution methods over a set of instances. Testing revealed parametric groups whose direct optimization performed poorly or could not identify a solution within a given time limit. Pareto front analysis of the solution methods showed which provided the greatest merit. The most feasible solutions were identified by first routing asset types that provide proximal service but doing so by imposing lower bounds on demand service windows, as informed by a Nearest Neighbor preprocessing technique. However, this technique degrades solution quality, so viable alternatives include bounds informed by a preprocessing technique based on the Floyd-Warshall algorithm. The easiest to solve instance remained best solved through direct optimization via a commercial solver.

Finally, an in-depth analysis of routing for multi-agent routing explored predetermined sequential order over a contested network. The last research section formulated

a bilevel problem, wherein the lower-level problem routes different asset types to satisfy demands while minimizing the cumulative service times. At the same time, the upper-level simultaneously seeks to identify a strategy that imposes a bounded number of disruptive actions that slow down agent travel on subsets of the network. To solve the bilevel problem, a greedy construction heuristic (GCH) found a baseline solution. After that, a simulated annealing metaheuristic (SA) and enhanced simulated annealing (eSA) leveraging a priority-based candidate solution identification and a tabu list of previously considered solutions explore the feasible solution space looking for a possible improvement. Over a set of developed test instances, SA and eSA improved upon GCH solutions, with the latter doing so more consistently.

In addition to recommendations and future research excursions mentioned in each research piece's respective section, further study is worth exploring. HATRH and CoVRP-PS methodologies do not include an adversary removing the temporary presence of an asset or limiting travel on arcs. More so, comparing the methodologies to real-world networks to see how performance is transcribed to fit realistic or historic scenarios, allowing for the complete connection between the research outlined and the desirable application.

Bibliography

Aksen, D. and Aras, N. (2013), 'A matheuristic for leader-follower games involving facility location-protection-interdiction decisions', *Metaheuristics for Bi-level Optimization* pp. 115–151.

Albers, S., Eilts, S., Even-Dar, E., Mansour, Y. and Roditty, L. (2014), 'On Nash equilibria for a network creation game', *ACM Transactions on Economics and Computation* **2**(1), 1–27.

Alcazar, V. (2012), 'Crisis management and the anti-access/area denial problem', *Strategic Studies Quarterly* **6**(4), 42–70.

Anbuudayasankar, S., Ganesh, K., Mohapatra, S. et al. (2016), *Models for practical routing problems in logistics*, Springer, New York, NY.

Arora, S. and Barak, B. (2009), *Computational complexity: a modern approach*, Cambridge University Press, Cambridge, UK.

Badri, M. A., Mortagy, A. K. and Alsayed, C. A. (1998), 'A multi-objective model for locating fire stations', *European Journal of Operational Research* **110**(2), 243–260.

Bard, J. F. (2013), *Practical bilevel optimization: algorithms and applications*, Vol. 30, Springer Science & Business Media, New York, NY.

Bard, J. F. and Falk, J. E. (1982), 'An explicit solution to the multi-level programming problem', *Computers & Operations Research* **9**(1), 77–100.

Bard, J. F. and Moore, J. T. (1992), 'An algorithm for the discrete bilevel programming problem', *Naval Research Logistics* **39**(3), 419–435.

Barrie, D. (2019), 'Anti-access/area denial: bursting the 'no-go' bubble?'.
URL: *https://www.iiss.org/blogs/military-balance/2019/04/anti-access-area-denial-russia-and-crimea*

Baxter, A., Lagerman, H. W. and Keskinocak, P. (2019), 'Quantitative modeling in disaster management: A literature review', *IBM Journal of Research and Development* **64**(1/2), 3–1.

Bazaraa, M. S., Jarvis, J. J. and Sherali, H. D. (2008), *Linear programming and network flows*, 4 edn, John Wiley & Sons, Hoboken, NJ.
URL: *https://www.wiley.com/en-us/Linear+Programming+and+Network+Flows,+4th+Edition-p-9780470462720*

Bialas, W. F. and Karwan, M. H. (1984), 'Two-level linear programming', *Management Science* **30**(8), 1004–1020.

Bräysy, O. and Gendreau, M. (2005*a*), 'Vehicle routing problem with time windows, Part I: Route construction and local search algorithms', *Transportation Science* **39**(1), 104–118.

Bräysy, O. and Gendreau, M. (2005*b*), 'Vehicle routing problem with time windows, Part II: Metaheuristics', *Transportation Science* **39**(1), 119–139.

Brown, G., Carlyle, M., Salmerón, J. and Wood, K. (2006), 'Defending critical infrastructure', *Interfaces* **36**(6), 530–544.

Cabot, A. (2018), 'Fortress Russia: How can NATO defeat moscow's A2/AD strategy and air defenses?'.
URL: *https://nationalinterest.org/blog/buzz/fortress-russia-how-can-nato-defeat-moscows-a2ad-strategy-and-air-defenses-35087*

Calvete, H. I., Gale, C. and Mateo, P. M. (2008), 'A new approach for solving linear bilevel problems using genetic algorithms', *European Journal of Operational Research* **188**(1), 14–28.

Ceselli, A., Righini, G. and Salani, M. (2009), 'A column generation algorithm for a rich vehicle-routing problem', *Transportation Science* **43**(1), 56–69.

Chang, K.-H., Hsiung, T.-Y. and Chang, T.-Y. (2022), 'Multi-commodity distribution under uncertainty in disaster response phase: Model, solution method, and an empirical study', *European Journal of Operational Research* .

Chen, C., Demir, E. and Huang, Y. (2021), 'An adaptive large neighborhood search heuristic for the vehicle routing problem with time windows and delivery robots', *European Journal of Operational Research* **294**(3), 1164–1180.

Chiang, W.-C. and Russell, R. A. (1996), 'Simulated annealing metaheuristics for the vehicle routing problem with time windows', *Annals of Operations Research* **63**, 3–27.

Church, R. and Murray, A. (2018), *Location Covering Models: History, Applications and Advancements*, Advances in Spatial Science, Springer International Publishing, New York, NY.

Clark, B., Patt, D. and Walton, T. A. (2021), 'Implementing decision-centric warfare: Elevating command and control to gain an optionality advantage'.
URL: *https://www.hudson.org/research/16729*

Cliff, R., Burles, M., Chase, M. S., Eaton, D. and Pollpeter, K. L. (2007), Entering the dragon's lair: Chinese antiaccess strategies and their implications for the united states, Technical report, RAND CORP SANTA MONICA CA.

Colby, E. (2019), 'How to win America's next war', *Foreign Policy* **5**.

Colson, B., Marcotte, P. and Savard, G. (2007), 'An overview of bilevel optimization', *Annals of Operations Research* **153**, 235–256.

Conn, A. R., Gould, N. I. and Toint, P. L. (2000), *Trust region methods*, SIAM, Philadelphia, PA.

Cooper, L. (1963), 'Location-allocation problems', *Operations Research* **11**(3), 331–343.

Cordeau, J.-F., Desaulniers, G., Desrosiers, J., Solomon, M. M. and Soumis, F. (2002), The VRP with time windows, *in* P. Toth and D. Vigo, eds, 'The Vehicle Routing Problem', SIAM, Philadelphia, PA, chapter 7, pp. 157–193.

Costello, J., Mattis, P. and McReynolds, J. (2016), 'Electronic warfare and the renaissance of Chinese information operations', *China's Evolving Military Strategy* pp. 173–213.

Czumaj, A., Krysta, P. and Vöcking, B. (2002), Selfish traffic allocation for server farms, *in* 'Proceedings of the thiry-fourth annual ACM symposium on Theory of Computing', pp. 287–296.

Dantzig, G. B. and Ramser, J. H. (1959), 'The truck dispatching problem', *Management Science* **6**(1), 80–91.

DARPA News (2017), 'Strategic technology office outlines vision for "mosaic warfare"'.
URL: *ht tp s: // ww w. do nc io .n av y. mi l/ (5u dz c1 55 ib dg ke 45 4e po ce 55) / CH IP S/ Ar ti cl eD et ai ls .a sp x? I D= 93 05*

DARPA Tiles Together a Vision of Mosaic Warfare (2020).
URL: *https://www.darpa.mil/work-with-us/darpa-tiles-together-a-vision-of-mosiac-warfare*

Dempe, S. (2002), *Foundations of bilevel programming*, Springer Science & Business Media, New York, NY.

Dempe, S., Kalashnikov, V. and Rıos-Mercado, R. Z. (2005), 'Discrete bilevel programming: Application to a natural gas cash-out problem', *European Journal of Operational Research* **166**(2), 469–488.

DeNegre, S. T. and Ralphs, T. K. (2009), A branch-and-cut algorithm for integer bilevel linear programs, *in* 'Operations research and cyber-infrastructure', Springer, pp. 65–78.

Deptula, D. A., Penney, H. R., Stutzriem, L. A. and Gunzinger, M. (2019), *Restoring America's Military Competitiveness: Mosaic Warfare*, Mitchell Institute for Airpower Studies, Arlington, VA.

Desrochers, M., Desrosiers, J. and Solomon, M. (1992), 'A new optimization algorithm for the vehicle routing problem with time windows', *Operations Research* **40**(2), 342–354.

Desrochers, M. and Laporte, G. (1991), 'Improvements and extensions to the miller-tucker-zemlin subtour elimination constraints', *Operations Research Letters* **10**(1), 27–36.

Dimitri, P. (1998), *Bertsekas Network Optimization: Continuous and Discrete Models*, Athena Scientific Publisher, Belmont, MA.

Dorigo, M. and Di Caro, G. (1999), Ant colony optimization: a new meta-heuristic, *in* 'Proceedings of the 1999 Congress on Evolutionary Computation', Vol. 2, IEEE, pp. 1470–1477.

Dougherty, C. M. (2019), *Why America Needs a New Way of War*, Center for a New American Security, Washington, DC.

Drezner, Z. and Hamacher, H. (2004), *Facility Location: Applications and Theory*, Springer Berlin Heidelberg, Germany.

Ehrgott, M. (2008), 'Multiobjective optimization', *Ai Magazine* **29**(4), 47–47.

Eiselt, H. A. and Marianov, V. (2012), 'Mobile phone tower location for survival after natural disasters', *European Journal of Operational Research* **216**(3), 563–572.

Engstrom, J. (2018), Systems confrontation and system destruction warfare: How the Chinese People's Liberation Army seeks to wage modern warfare, Technical report, RAND Corporation Santa Monica United States.

Fabrikant, A., Luthra, A., Maneva, E., Papadimitriou, C. H. and Shenker, S. (2003), On a network creation game, *in* 'Proceedings of the twenty-second annual symposium on Principles of Distributed Computing', pp. 347–351.

Farahani, R. Z., Asgari, N., Heidari, N., Hosseininia, M. and Goh, M. (2012), 'Covering problems in facility location: A review', *Computers & Industrial Engineering* **62**(1), 368–407.

Farahani, R. Z., Hekmatfar, M., Fahimnia, B. and Kazemzadeh, N. (2014), 'Hierarchical facility location problem: Models, classifications, techniques, and applications', *Computers & Industrial Engineering* **68**, 104–117.

Ferber, J. and Weiss, G. (1999), *Multi-agent systems: an introduction to distributed artificial intelligence*, Vol. 1, Addison-Wesley Reading, Boston, MA.

Fix, E. and Hodges Jr, J. L. (1952), Discriminatory analysis-nonparametric discrimination: Small sample performance, Technical Report Project 21-49-004, Report Number 11, Air University School of Aviation Medicine, Randolph Field, TX.
URL: *https://apps.dtic.mil/sti/pdfs/ADA800391.pdf*

Floyd, R. W. (1962), 'Algorithm 97: Shortest path', *Communications of the ACM* **5**(6), 345.

Frank, M. (1981), 'The Braess Paradox', *Mathematical Programming* **20**(1), 283–302.

Fravel, M. T. (2016), 'China's changing approach to military strategy: The science of military strategy from 2001 and 2013', *The Evolution of China's Military Strategy (Washington, DC: Brookings Forthcoming), MIT Political Science Department Research Paper* .

Frey, C. M., Jungwirth, A., Frey, M. and Kolisch, R. (2022), 'The vehicle routing problem with time windows and flexible delivery locations', *European Journal of Operational Research* .

Glover, F. (1986), 'Future paths for integer programming and links to artificial intelligence', *Computers & Operations Research* **13**(5), 533–549.

Goel, A. (2010), A column generation heuristic for the general vehicle routing problem, *in* C. Blum and R. Battiti, eds, 'LION 2010: International Conference on Learning and Intelligent Optimization', Springer, Berlin, pp. 1–9.

Goel, A. and Gruhn, V. (2008), 'A general vehicle routing problem', *European Journal of Operational Research* **191**(3), 650–660.

Golden, B. L., Raghavan, S. and Wasil, E. A. (2008), *The vehicle routing problem: latest advances and new challenges*, Springer, New York, NY.

Golden, B., Wang, X. and Wasil, E. (2023), The evolution of the vehicle routing problem—a survey of vrp research and practice from 2005 to 2022, *in* 'The Evolution of the Vehicle Routing Problem', Springer, New York, NY, pp. 1–64.

Hakimi, S. L. (1965), 'Optimum distribution of switching centers in a communication network and some related graph theoretic problems', *Operations Research* **13**(3), 462–475.

Hoehn, J. R. (2022), Advanced battle management system (abms), Technical report, Congressional Research Service In Focus.

Holland, J. H. (1973), 'Genetic algorithms and the optimal allocation of trials', *SIAM Journal on Computing* **2**(2), 88–105.

Ishizuka, Y. and Aiyoshi, E. (1992), 'Double penalty method for bilevel optimization problems', *Annals of Operations Research* **34**(1), 73–88.

Israeli, E. and Wood, R. K. (2002), 'Shortest-path network interdiction', *Networks: An International Journal* **40**(2), 97–111.

Jane's (2021), *All the worlds aircraft: development & production*, Jane's, Coulsdon, United Kingdom.

Janjarassuk, U. and Nakrachata-Amon, T. (2015), A simulated annealing algorithm to the stochastic network interdiction problem, *in* '2015 IEEE International Conference on Industrial Engineering and Engineering Management', IEEE, pp. 230–233.

Jayarathna, N., Lanel, J. and Juman, Z. (2020), 'Five years of multi-depot vehicle routing problems', *Journal of Sustainable Development of Transport and Logistics* **5**(2), 109–123.

Jeroslow, R. G. (1985), 'The polynomial hierarchy and a simple model for competitive analysis', *Mathematical Programming* **32**(2), 146–164.

Jouzdani, J., Sadjadi, S. J. and Fathian, M. (2013), 'Dynamic dairy facility location and supply chain planning under traffic congestion and demand uncertainty: A case study of tehran', *Applied Mathematical Modelling* **37**(18-19), 8467–8483.

Karp, R. M. (1972), Reducibility among combinatorial problems, *in* 'Complexity of computer computations', Springer, New York, NY, pp. 85–103.

Karp, R. M. and Papadimitriou, C. H. (1982), 'On linear characterizations of combinatorial optimization problems', *SIAM Journal on Computing* **11**(4), 620–632.

Kass, L. (2019), 'US air power: The imperative for modernization (buy the f-35)'. **URL:** *https://breakingdefense.com/2019/03/us-air-power-the-imperative-for-modernization-buy-the-f-35/*

Kennedy, J. and Eberhart, R. (1995), Particle swarm optimization, *in* 'Proceedings of ICNN'95-International Conference on Neural Networks', Vol. 4, IEEE, pp. 1942–1948.

Kirkpatrick, S., Gelatt Jr, C. D. and Vecchi, M. P. (1983), 'Optimization by simulated annealing', *Science* **220**(4598), 671–680.

Koutsoupias, E. and Papadimitriou, C. (2009), 'Worst-case equilibria', *Computer science Review* **3**(2), 65–69.

Kuo, R.-J., Lee, Y., Zulvia, F. E. and Tien, F. (2015), 'Solving bi-level linear programming problem through hybrid of immune genetic algorithm and particle swarm optimization algorithm', *Applied Mathematics and Computation* **266**, 1013–1026.

Kuo, R., Lu, S.-H., Lai, P.-Y. and Mara, S. T. W. (2022), 'Vehicle routing problem with drones considering time windows', *Expert Systems with Applications* **191**, 116264.

LaGrone, S. (2021), 'Milley: China wants capability to take Taiwan by 2027, sees no near-term intent to invade'.
URL: *https://news.usni.org/2021/06/23/milley-china-wants-capability-to-take-taiwan-by-2027-sees-no-near-term-intent-to-invade*

Lei, X., Shen, S. and Song, Y. (2018), 'Stochastic maximum flow interdiction problems under heterogeneous risk preferences', *Computers & Operations Research* **90**, 97–109.

Lenstra, J. K. and Kan, A. R. (1976), 'On general routing problems', *Networks* **6**(3), 273–280.

Lessin, A. M., Lunday, B. J. and Hill, R. R. (2018), 'A bilevel exposure-oriented sensor location problem for border security', *Computers & Operations Research* **98**, 56–68.

Li, H., Chen, J., Wang, F. and Bai, M. (2021), 'Ground-vehicle and unmanned-aerial-vehicle routing problems from two-echelon scheme perspective: A review', *European Journal of Operational Research* **294**(3), 1078–1095.

Liu, B., Sheu, J.-B., Zhao, X., Chen, Y. and Zhang, W. (2020), 'Decision making on post-disaster rescue routing problems from the rescue efficiency perspective', *European Journal of Operational Research* **286**(1), 321–335.

Luis, E., Dolinskaya, I. S. and Smilowitz, K. R. (2012), 'Disaster relief routing: Integrating research and practice', *Socio-economic Planning Sciences* **46**(1), 88–97.

Lunday, B. J. and Sherali, H. D. (2012*a*), 'Minimizing the maximum network flow: models and algorithms with resource synergy considerations', *Journal of the Operational Research Society* **63**, 1693–1707.

Lunday, B. J. and Sherali, H. D. (2012*b*), 'Network interdiction to minimize the maximum probability of evasion with synergy between applied resources', *Annals of Operations Research* **196**, 411–442.

Lunday, B. J., Sherali, H. D. and Lunday, K. E. (2012), 'The coastal seaspace patrol sector design and allocation problem', *Computational Management Science* **9**(4), 483–514.

Ma, M., Huang, H., Song, X., Peña-Mora, F., Zhang, Z. and Chen, J. (2022), 'Optimal sizing and operations of shared energy storage systems in distribution networks: A bi-level programming approach', *Applied Energy* **307**, 118170.

Magnuson, S. (2018), 'Darpa pushes 'Mosaic Warfare' concept'.
URL: *https://www.nationaldefensemagazine.org/articles/2018/11/16/darpa-pushes-mosaic-warfare-concept*

Marques-Silva, J. P. and Sakallah, K. A. (1999), 'Grasp: A search algorithm for propositional satisfiability', *IEEE Transactions on Computers* **48**(5), 506–521.

Mattis, J. (2018), *Summary of the 2018 National Defense Strategy of the United States of America*, Department of Defense, Washington, DC.

Michalopoulos, D. P., Barnes, J. W. and Morton, D. P. (2015), 'Prioritized interdiction of nuclear smuggling via tabu search', *Optimization Letters* **9**, 1477–1494.

Miehle, W. (1958), 'Link-length minimization in networks', *Operations Research* **6**(2), 232–243.

Miller, C. E., Tucker, A. W. and Zemlin, R. A. (1960), 'Integer programming formulation of traveling salesman problems', *Journal of the Association for Computing Machinery* **7**(4), 326–329.

Moore, J. T. and Bard, J. F. (1990), 'The mixed integer linear bilevel programming problem', *Operations Research* **38**(5), 911–921.

Mulvenon, J. C., Tanner, M. S., Chase, M. S., Frelinger, D., Gompert, D. C., Libicki, M. C. and Pollpeter, K. L. (2006), *Option Four: Chinese Network-Centric Warfare*, 1 edn, RAND Corporation, chapter 7, pp. 133–144.

Nadizadeh, A., Sahraeian, R., Zadeh, A. S. and Homayouni, S. M. (2011), 'Using greedy clustering method to solve capacitated location-routing problem', *African Journal of Business Management* **5**(21), 8470–8477.

Nallusamy, R., Duraiswamy, K., Dhanalaksmi, R. and Parthiban, P. (2010), 'Optimization of multiple vehicle routing problems using approximation algorithms', *arXiv preprint arXiv:1001.4197* .

Neagoe, V. and Borşa, S.-S. (2019), 'Anti-access/area denial strategy–conventional war, hybrid war or asymmetric war?', *Strategic Impact* **70/71**, 15–20.
URL: *https://www.ceeol.com/search/article-detail?id=853247*

Nelder, J. A. and Mead, R. (1965), 'A simplex method for function minimization', *The Computer Journal* **7**(4), 308–313.

Nemhauser, D. B. G. and Wolsey, L. (1993), *Integer programming and combinatorial optimization*, Springer, New York, NY.

Ochmanek, D. A. (2022), Determining the military capabilities most needed to counter china and russia: A strategy-driven approach, Technical report, RAND CORP SANTA MONICA CA.

O'Donoughue, N. A., McBirney, S. and Persons, B. (2021), Distributed kill chains: Drawing insights for Mosaic Warfare from the immune system and from the navy, Technical report, Rand Corporation Arlington.

Osman, I. H. (1993), 'Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem', *Annals of Operations Research* **41**, 421–451.

Ostrowski, J., Linderoth, J., Rossi, F. and Smriglio, S. (2011), 'Orbital branching', *Mathematical Programming* **126**, 147–178.

Parsafard, M. and Li, X. (2021), 'Sensor location design for interdicting mobile travelers with probabilistic space-time trajectories', *Transportation Research Part C: Emerging Technologies* **132**, 103420.

Perakis, G. and Roels, G. (2007), 'The price of anarchy in supply chains: Quantifying the efficiency of price-only contracts', *Management Science* **53**(8), 1249–1268.

Perea, F. and Puerto, J. (2013), 'Revisiting a game theoretic framework for the robust railway network design against intentional attacks', *European Journal of Operational Research* **226**(2), 286–292.

Photonics Media (2019), 'Piecing together the future battlefield with Mosaic Warfare at DCS'.
URL: *https://www.photonics.com/Articles/Piecing_Together_the_Future_Battlefield_with_/a64571*

Pickrell, R. (2019), 'The US has been getting 'its ass handed to it' in simulated war games against Russia and China, analysts say', *Task & Purpose* **8**.
URL: *https://taskandpurpose.com/news/russia-china-war-games/*

Robbins, M. J. and Lunday, B. J. (2016), 'A bilevel formulation of the pediatric vaccine pricing problem', *European Journal of Operational Research* **248**(2), 634–645.

Roughgarden, T. (2009), Intrinsic robustness of the price of anarchy, *in* 'Proceedings of the forty-first annual ACM symposium on Theory of Computing', pp. 513–522.

Sadati, M. E. H., Aksen, D. and Aras, N. (2020*a*), 'The r-interdiction selective multi-depot vehicle routing problem', *International Transactions in Operational Research* **27**(2), 835–866.

Sadati, M. E. H., Aksen, D. and Aras, N. (2020*b*), 'A trilevel r-interdiction selective multi-depot vehicle routing problem with depot protection', *Computers & Operations Research* **123**, 104996.

Sahin, K. H. and Ciric, A. R. (1998), 'A dual temperature simulated annealing approach for solving bilevel programming problems', *Computers & Chemical Engineering* **23**(1), 11–25.

Santillán, C. G., Reyes, L. C., Rodríguez, M. L. M., Barbosa, J. J. G., López, O. C., Zarate, G. R. and Hernández, P. (2012), Variants of VRP to optimize logistics

management problems, *in* 'Logistics Management and Optimization through Hybrid Artificial Intelligence Systems', IGI Global, pp. 207–237.

Sapaty, P. S. (2019), 'Mosaic warfare: From philosophy to model to solution', *Institute of Mathematical Machines and Systems Problems National Academy of Sciences of Ukraine* **3**(3), 17–34.

Schilling, D. A. (1993), 'A review of covering problems in facility location', *Location Science* **1**, 25–55.

Schwartz, M. (2010), 'Defense acquisitions: How DoD acquires weapon systems and recent efforts to reform the process'.

Shoham, Y. and Leyton-Brown, K. (2008), *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*, Cambridge University Press, Cambridge, United Kingdom.

Sluijk, N., Florio, A. M., Kinable, J., Dellaert, N. and Van Woensel, T. (2022), 'Two-echelon vehicle routing problems: A literature review', *European Journal of Operational Research* .

Smith, J. C. and Song, Y. (2020), 'A survey of network interdiction models and algorithms', *European Journal of Operational Research* **283**(3), 797–811.

Stackelberg, H. v. (1952), *Theory of the market economy*, Oxford University Press.

Starita, S. and Scaparra, M. P. (2021), 'Assessing road network vulnerability: A user equilibrium interdiction model', *Journal of the Operational Research Society* **72**(7), 1648–1663.

Szmigiera, M. (2022), 'Ranking: Military spending by country 2021'.
URL: *https://www.statista.com/statistics/262742/countries-with-the-highest-military-spending*

Thakoor, O., Garg, J. and Nagi, R. (2019), 'Multiagent UAV routing: A game theory analysis with tight price of anarchy bounds', *IEEE Transactions on Automation Science and Engineering* **17**(1), 100–116.

Tilk, C., Olkis, K. and Irnich, S. (2021), 'The last-mile vehicle routing problem with delivery options', *OR Spectrum* **43**(4), 877–904.

Tirpak, J. A. (2000), 'Find, fix, track, target, engage, assess'.
URL: *https://www.airforcemag.com/article/0700find/*

Toth, P. and Vigo, D. (2002), *The Vehicle Routing Problem (Monographs on Discrete Mathematics and Applications, Series Number 9*, SIAM, Philadelpia, Pennsylvania.

Toth, P. and Vigo, D. (2014), *Vehicle routing: problems, methods, and applications*, SIAM, Philadelphia, PA.

United States Joint Chiefs of Staff (2021), *Joint Publication 3-30: Joint Air Operations*, Washington, DC.

U.S. Air Force (2014), 'RQ-4 Global Hawk'.
URL: *https://www.af.mil/About-Us/Fact-Sheets/Display/Article/104516/rq-4-global-hawk/*

U.S. Air Force (2015), 'E-3 Sentry (AWACS)'.
URL: *https://www.af.mil/About-Us/Fact-Sheets/Display/Article/104504/e-3-sentry-awacs/*

U.S. Air Force (2017), 'Joint Direct Attack Munition GBU- 31/32/38'.
URL: *https://www.af.mil/About-Us/Fact-Sheets/Display/Article/104572/joint-direct-attack-munition-gbu-313238/*

U.S. Air Force (2021), 'F-16 Fighting Falcon'.
URL: *https://www.af.mil/About-Us/Fact-Sheets/Display/Article/104505/f-16-fighting-falcon/*

Van Breedam, A. (1995), 'Improvement heuristics for the vehicle routing problem based on simulated annealing', *European Journal of Operational Research* **86**(3), 480–490.

Verter, V. and Dasci, A. (2002), 'The plant location and flexible technology acquisition problem', *European Journal of Operational Research* **136**(2), 366–382.

Vincent, F. Y., Redi, A. P., Hidayat, Y. A. and Wibowo, O. J. (2017), 'A simulated annealing heuristic for the hybrid vehicle routing problem', *Applied Soft Computing* **53**, 119–132.

Wang, G., Wang, X., Wan, Z. and Lv, Y. (2007), 'A globally convergent algorithm for a class of bilevel nonlinear programming problem', *Applied Mathematics and Computation* **188**(1), 166–172.

Weber, A. (1909), *Theory of the location of industries*, University of Chicago Press, Chicago, IL.

Wood, R. K. (1993), 'Deterministic network interdiction', *Mathematical and Computer Modelling* **17**(2), 1–18.

Wu, L.-Y., Zhang, X.-S. and Zhang, J.-L. (2006), 'Capacitated facility location problem with general setup cost', *Computers & Operations Research* **33**(5), 1226–1241.

Yabuta, K. and Kitazawa, H. (2008), Optimum camera placement considering camera specification for security monitoring, *in* 'IEEE International Symposium on Circuits and Systems', pp. 2114–2117.

Ye, D., Chen, L. and Zhang, G. (2021), 'On the price of anarchy of two-stage machine scheduling games', *Journal of Combinatorial Optimization* **42**(3), 616–635.

Yevtodyeva, M. (2022), 'Development of the chinese a2/ad system in the context of us–china relations', *Herald of the Russian Academy of Sciences* **92**(Suppl 6), S534–S542.

Yin, Y. (2000), 'Genetic-algorithms-based approach for bilevel programming models', *Journal of Transportation Engineering* **126**(2), 115–120.

Yousefi, A. and Donohue, G. (2004), Temporal and spatial distribution of airspace complexity for air traffic controller workload-based sectorization, *in* 'AIAA 4th Aviation Technology, Integration and Operations (ATIO) Forum', p. 6455.

Zhang, B., Zhao, M. and Hu, X. (2022), 'Location planning of electric vehicle charging station with users' preferences and waiting time: Multi-objective bi-level programming model and HNSGA-II algorithm', *International Journal of Production Research* pp. 1–30.

Zokaee, S., Bozorgi-Amiri, A. and Sadjadi, S. J. (2016), 'A robust optimization model for humanitarian relief chain design under uncertainty', *Applied Mathematical Modelling* **40**(17-18), 7996–8016.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704–0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202–4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
14–09–2023	Dissertation	March 2021 — September 2023

4. TITLE AND SUBTITLE

Analysis of Multi-agent Routing Solution Methodologies Exploring a Mosaic Warfare Strategy

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Donnel, Stephen D., Capt, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENS-DS-23-S-014

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Strategic Development Planning & Experimentation (SDPE) Office
Mr. David J. Myers
1864 4th Street
Wright-Patterson AFB, OH 45433
(937) 904-6539

10. SPONSOR/MONITOR'S ACRONYM(S)

SDPE

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Distribution Statement A. Approved for Public Release; Distribution Unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Recognizing that communication between assets may be possible locally but not globally (e.g., due to disruptions to a communication network), Mosaic Warfare requires the movement and operation of multiple, dispersed assets in smaller groups (i.e., *tiles*), within which exist hierarchical, functional relationships between assets. This research first evaluates a heuristic for an enterprise of aerial assets comprised of airborne sensors, command and control, and strike aircraft seeking to move towards and destroy stationary targets. Next, we examine routing multiple assets of different types over a network to service demands in a collaborative manner, in that, when servicing a demand, the differing asset types must do so nearly simultaneously. Finally, research explores routing multiple assets of different types over a network to service demands in sequential order. Moreover, we seek to identify effective network disruption strategies with limited resources to maximize the minimal cumulative service time. Within a bilevel programming structure for this Stackelberg game, the upper-level problem determines the disruption strategy, and the lower-level problem routes the assets.

15. SUBJECT TERMS

Multi-agent Routing, Hierarchical Assets, Mosaic Warfare, Adjustable Time Windows, Proximal Service, Model Decomposition, Feasible Region Reduction, Bilevel Programming, Network Interdiction, Simulated Annealing, Game Theory

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Brian J. Lunday, AFIT/ENS
U	U	U	UU	155	19b. TELEPHONE NUMBER *(include area code)* (937)-255-3636, x4624; Brian.Lunday@afit.edu