ARL-MR-1114 ● Nov 2024

Automated Task Allocation Models for Armored Platoons

by Mark Dranias, Angelique Scharine, Alfred Yu, Sarah Al-Hussaini, Craig Johnson, Evan Carter, Laura Marusich, Ahmed Khalil, Yoonjae Lee, Efstathios Bakolas, and Gregory Gremillion

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-MR-1114 ● Nov 2024

Automated Task Allocation Models for Armored Platoons

Mark Dranias
DCS Corporation

Angelique Scharine, Alfred Yu, Craig Johnson,
Evan Carter, Laura Marusich, and Gregory Gremillion
DEVCOM Army Research Laboratory

Sarah Al-Hussaini
Army Educational Outreach Program

Ahmed Khalil, Yoonjae Lee, and Efstathios Bakolas
The University of Texas at Austin

REPORT DOCUMENTATION PAGE

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
November 2024	Memorandum Report	1 October 2023	30 September 2024

4. TITLE AND SUBTITLE
Automated Task Allocation Models for Armored Platoons

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
5a. CONTRACT NUMBER University of Texas-Austin: CA1: Contract No. W911NF-21-2-0043 CA2: Contract No. W911NF-22-2-0091 Georgia Tech: Contract No. W911NF-20-2-0036		
5d. PROJECT NUMBER	**5e. TASK NUMBER**	**5f. WORK UNIT NUMBER**

6. AUTHOR(S)
Mark Dranias, Angelique Scharine, Alfred Yu, Sarah Al-Hussaini, Craig Johnson, Evan Carter, Laura Marusich, Ahmed Khalil, Yoonjae Lee, Efstathios Bakolas, and Gregory Gremillion

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
DEVCOM Army Research Laboratory ATTN: FCDD-RLA-FD Aberdeen Proving Ground, MD	ARL-MR-1114

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited.

13. SUPPLEMENTARY NOTES
ORCIDs: Mark Dranias, 0000-0003-4615-1069; Angelique Scharine, 0000-0001-8230-1824; Alfred Yu, 0000-0003-4531-3206; Sarah Al-Hussaini, 0000-0003-4175-4506; Craig Johnson, 0000-0003-3739-9679; Evan Carter, 0000-0001-7471-8769; Laura Cooper, 0000-0002-3524-6110; Gregory Gremillion, 0000-0002-0205-688X; Ahmed Khalil, 0009-0009-4859-8201; Yoonjae Lee, 0000-0003-2953-8907; Efstathios Bakolas, 0000-0003-4104-0108

14. ABSTRACT
Although armored platoon crew members have typically had defined and static roles, the next generation of combat vehicle, with reduced Soldier-to-system ratio, partially manned formations, and increased integration of autonomous agents, will require a more flexible approach to resource allocation and task assignment. Several models have been developed to automate and optimize this task assignment using a variety of approaches. Here we implement two approaches: one that makes assignments empirically, learned from expert demonstrations; and another that optimizes performance by maximizing the value of the assignment using a priori-defined values. Both approaches have their limitations, but we demonstrate how they might work in this future application space and discuss their implications.

15. SUBJECT TERMS
decision-making, task allocation, multi-agent planning, resource allocation, game-theoretical, wonderful life utility, Humans in Complex Systems

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	C. THIS PAGE		
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UU	59

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
Angelique Scharine	(410) 278-5957

STANDARD FORM 298 (REV. 5/2020)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

List of Tables

1. Introduction

Armored vehicles, such as the M1-series Abrams tank, are considered a vital component of a modern combined-arms military as they can provide mobile, protected firepower on the battlefield. However, current trends suggest a future environment characterized by new challenges and increasing levels of complexity, dynamism, and geographic distribution. New armored platforms are being developed to address the challenges of the future battlefield and reduce risk to operators, while filling a core role in multidomain operations.

Under current Army doctrine (Headquarters, Department of the Army 2019), a tank platoon consists of four M1A2 tanks and their crews. The tank platoon can be further subdivided into two subteams ("sections") of two tanks. Each tank crew consists of four Soldiers performing separate but fixed functions, including a tank commander (who may also be a platoon leader or platoon sergeant), gunner, loader, and driver for a total of 16 Soldiers in the platoon. It is proposed that the next-generation combat vehicle (NGCV) platoon will consist of six armored vehicles, which are further subdivided into two sections of three vehicles including one manned combat vehicle (MCV), and two robotic combat vehicles (RCVs) each. In addition to the six ground vehicles, each vehicle may have one unmanned aerial vehicle (UAV) asset. Each MCV would hold a section of seven Soldiers, including a platoon leader or platoon sergeant and three driver–gunner pairs who each operate the MCV or one of the two RCVs. In brief, the number of vehicles in a platoon is increasing (from four to six, or more if UAVs are considered) while the number of Soldiers in that platoon is decreasing (from 16 to 14). This reduction in Soldier-to-system ratio necessitates a greater span of control for each crew member.

It is hoped that many tasks will become assisted or automated with the advent of the Warfighter machine interface (WMI), which provides an enhanced common operating picture (COP) and utilizes multiple sensors to provide indirect vision, autonomous navigation capabilities, and automatic target recognition. In contrast to legacy platforms, the combination of the WMI and interoperable crew stations allows for dynamic tasking wherein Soldiers can operate any of the vehicles, UAVs, or weapons. This allows for the crews to flexibly adjust their human–autonomy team configuration on the fly to adapt to different situations to a degree that is not possible in modern tanks. Fundamentally, human and machine resources within the platoon could be retasked dynamically between vehicles and subsystems when needed, based on changing mission contexts.

The commander/leader's role is to assess the resources at hand such as vehicle status, fuel, ammunition, and personnel, and assign tasks based on the current

tactical situation according to priorities and standard operating procedures (SOPs). In a modern tank crew, this role is critical for the collective function of the tank because the roles are extremely interdependent. However, more tasks will need to be handled by fewer people in the NGCV environment, and the coordination of those tasks will be more complex due to dynamic retasking capabilities.

1.1 Problem

There is a reduction in personnel in a NGCV platoon while there is an increase in the number of vehicles. It falls to the commander to maintain awareness of the resources available to his team and to optimize performance through task assignment and reassignment as the context changes. This adds to the cognitive load of the commander and, depending on the concurrent events and tasks required of the commander, may lead to errors or detract from the commander's performance. This ultimately will have a negative effect on the performance of the platoon as a collective, impacting their safety and effectiveness. Therefore, automated and algorithmic capabilities that assist with quickly assessing these situations and generating a proposed reallocation of these resources are critical for leverage of their full combat power, limiting the burden on the decision-maker and responsiveness of the platoon. This report outlines developments in two approaches for using algorithms to manage dynamic retasking.

1.2 Proposed Approach

There have been efforts to develop models to autonomously assign tasks based on resources, priorities, and SOPs. First, we outline some of these approaches, and then we present two different models and demonstrate how they might be implemented in our operational environment. The two approaches differ as the first requires demonstration data for model training and aims to replicate a demonstrated assignment strategy while the second assumes that there exists some a priori knowledge of the utilities associated with each possible pairing of agent to task, where the optimal assignment is the one that results in the highest overall utility.

1.3 Task Allocation

The problem of task allocation for an armored platoon commander is that of coalition formation. A group of agents, consisting of autonomous software agents and human Soldiers in this case, must work together to achieve a series of tasks. Each agent has specific capabilities and goals, and they must work together to solve a problem that has spatial and temporal constraints. Jones et al. (2011) stated that the challenges of coalition formation with spatial and temporal constraints problem

(CFSTP) consist of finding optimal teams, adapting to a dynamically changing environment, and efficient use of heterogeneous agents with different capabilities.

1.3.1 Models of Multi-Agent Planning (MAP)

If task requirements are clearly defined, and each asset's probability of success known, it should be possible to arrive at a mathematically optimal solution to the MAP problem. However, the problem is rarely this simple. As the complexity of the problem increases, the decision-maker, algorithm, or human must consider all aspects of the problem prior to making assignments. In practice, we often rely on heuristics developed from training and experience. Good leaders are supposed to be able to do this quickly and under pressure. This presumes, however, that SOPs represent empirically proven best practices and that it is possible to consider all relevant factors. Ideally, an algorithmic decision-making aid could go beyond default procedures to suggest superior alternatives and, by doing so, maximize the effectiveness of those resources and unburden leaders during times of high workload.

1.3.1.1 Centralized versus Decentralized Decision-Making

Decisions can either be determined centrally by a leader who makes assignment in a top-down manner or by the individual agents in a distributed manner (Gerkey and Matarić 2004). In the former case, it is presumed that the leader is aware of the capabilities of each agent and the requirements for the overall objective. Agents negotiate with each other in a distributed or decentralized system, sharing information about task requirements and agents' capabilities. While centrally located decision-making may ensure that all factors are known and considered by the decision-maker, it can become a bottleneck and make decision-making slow if a single entity is responsible for all decisions, especially as the number of agents and environmental variables increases. Conversely, distributed decision-making can be problematic, as individual agents may not be aware of all relevant factors related to a task. Furthermore, distributed decision-making requires more communication among agents as they negotiate responsibility for assignments (Macarthur et al. 2011).

Whether centrally or distributed, it is necessary to specify how decisions will be made. A common framework used in MAP models uses a market or auction mechanism. Agents bid on task assignments based on their knowledge of their own capabilities and the task requirements (Sarne and Krause 2005; Zlot 2006; Nanjanath and Gini 2010; Ye et al. 2024). Assignments are given to the agent with the best capabilities for a task. In a centralized model, the agents make bids to an "auctioneer" who makes the final decision (Gerkey and Matarić 2002). In contrast,

agents negotiate directly with each other in a decentralized model (Choi et al. 2009; Nanjanath and Gini 2010; Capitan et al. 2013) and make decisions based on local information and communication (Shorinwa et al. 2020; Hirsch et al. 2021; Mohammad et al. 2023).

An alternative to the auction method, the "wonderful life utility" (WLU) function reduces the cost of communication among agents because it does not require direct communication among agents. Instead, individual agents are rewarded based on how well their actions achieve a larger goal (Wolpert et al. 1999). The WLU function essentially calculates the difference between how well the whole system does with the agent and how well it would do without the agent. This way, agents are incentivized to take actions that benefit the entire system, not just themselves.

1.3.1.2 Static versus Dynamic Sequencing

Another consideration is how the model handles sequencing of tasks and whether the approach is static or dynamic. Static planning assumes that nothing will change after the plan is set in action and, in fact, all decisions will be made before any initial steps are taken. Dynamic planning assumes that plans will change as events alter the capabilities and objectives of the team. For example, planning state machines treat stages in an overall plan as individual states for which there are transitions and actions generated by a central planning engine (Tozicka et al. 2015). Although possible outcomes are considered during planning, plans are made prior to initiation of the tasks.

In the forward MAP (FMAP) model, potential future states are considered during planning and contingent plans are developed for each alternative (Nissim and Brafman 2014; Torreno et al. 2014). FMAP also allows partial order planning, meaning that initial steps can be taken before the entire plan has been made, allowing for the flexibility to adapt to changing conditions (Boutilier and Brafman 2001).

1.3.2 The NGCV Environment

Typically, decision-making for collective action in Army platoons is the responsibility of the tank commander (at the individual vehicle level) or platoon leader (at the entire platoon level). That said, each Soldier in the platoon is expected to understand the tasks they are responsible for and is given some latitude in deciding how to achieve them. Furthermore, they are expected to take initiative for adjusting as needed, with the consent of their commander. The decision-making responsibilities placed upon the tank commanders and platoon leader can potentially result in coordination bottlenecks, wherein overload on that one person can negatively impact the performance of the entire crew or platoon (Stanton and

Roberts 2020). Automation of task allocation is intended to remove some of the bottleneck surrounding having a single decision-maker, informing crew members of the tasks requiring agent resources, resulting in more fluid coordination, allocation, and execution of tasks.

The military operational environment changes dynamically and the available agents and task requirements of future states is not known in advance. Consequently, the models presented here presume that task allocation is static and only remains in effect until either the task or the resources change to a sufficiently large degree that the prior allocation is no longer valid or optimal. At that point, the model must reallocate agents to tasks based on the new circumstances.

Of the two approaches considered here, one requires data upon which to derive model parameters and the other requires defining the value associated with each potential assignment, assuming the optimal set of assignments can be computed as those that provide the highest overall value. In truth, there is no guarantee that demonstration data will reflect best practices in task assignment, nor that the utility of task assignments is known, understood, or sufficiently quantified.

1.4 Data-Driven Allocation Model

At the Georgia Institute of Technology (GT), Srikanthan and Ravichandar (2022) proposed a data-driven model for resource-aware use of heterogeneous strategies for coalition formation. They suggested a two-part approach to improve coalition formation in teams with diverse skillsets (heterogeneous teams).

First, the approach learns different strategies for forming coalitions by observing how human experts tackle similar tasks. Unlike existing methods that require precise task specifications, this approach can learn even from implicit strategies used by the experts. This is the reinforcement learning part of the model; its strategies are based on those of subject-matter experts.

Secondly, the approach utilizes a resource-aware adaptation technique. It analyzes the available resources (skills of the agents) and selects the most fitting strategy from the learned ones to form coalitions for the specific team. This allows for adaptation without needing additional training for each new scenario.

Overall, this approach aims at creating more flexible coalition formation in multi-agent systems by considering both previously successful strategies and the specific resources available in the current team.

1.5 Optimization-Driven Allocation Model

At the University of Texas at Austin (UT), Lee et al. (2025) proposed a cost-optimization model that presumes certain assignments have higher value than others and aims to optimize assignments while operating within certain constraints. Thus, there is an assumption that preferred operational practices are known (a priori). This allows the model to incorporate SOPs while allowing for deviation from those practices when they result in a higher utility value. For example, a platoon may begin a mission operating according to SOPs, but as assets become unavailable, retasking allows the teams to retain the most value.

2. Resource-Aware Allocation Model

There are two levels to consider when applying this model to the mounted platoon: high-level and low-level. The high-level formulation of the problem addresses the problem of assigning vehicles to concurrent tasks. The low-level problem addresses how to assign crew members and autonomy to vehicle subsystems (e.g., mobility, lethality, slewable sensor).

The GT model relies on a theoretical approach that is general enough to be applied to either problem. The general case of the model is reviewed and then adapted for application to the low-level (FY22) and high-level (FY24) problems.

2.1 Overview of GT-Resource Aware Allocation (RSA) Algorithm and Theory

GT formulates the task assignment problem as assigning a heterogenous team of S species of robots, in which the s^{th} species contains N_s robots to a set of M concurrent tasks denoted by $T = \{T_1, T_2, \ldots, T_M\}$. Figure 1 shows the structure of the matrices used to solve this problem.

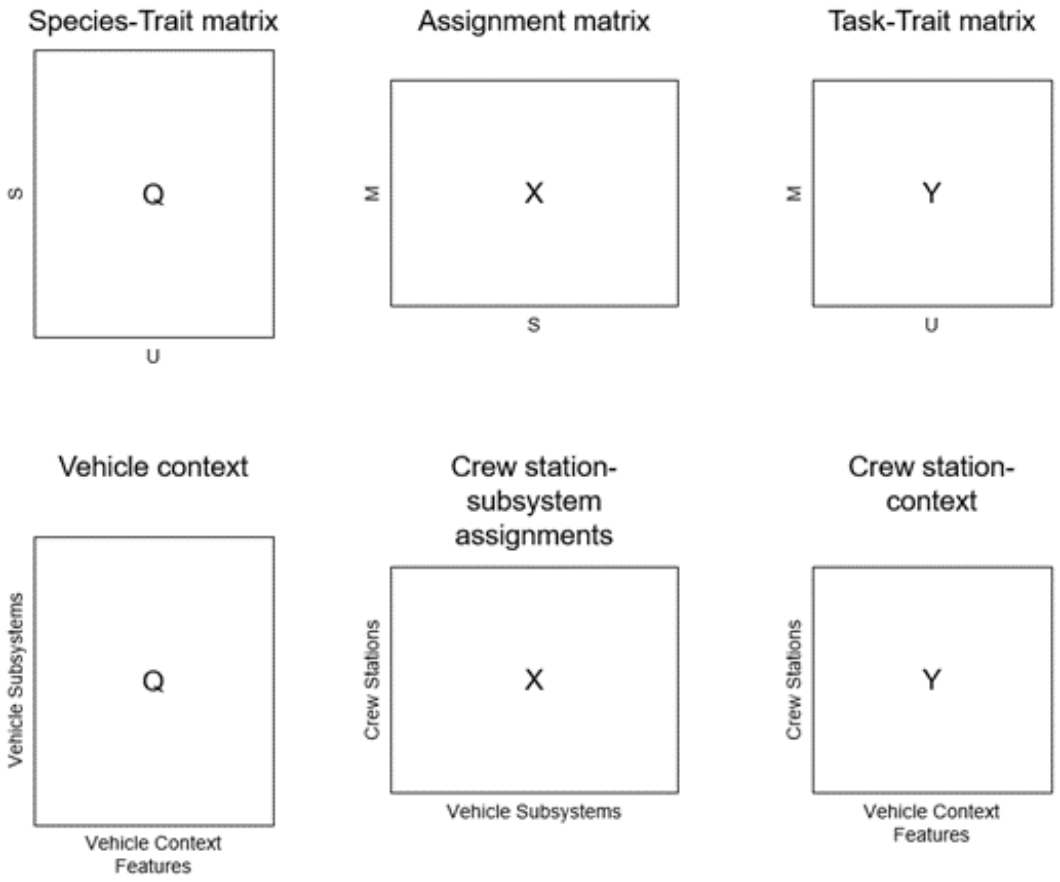

Fig. 1 Matrices and matrix components as applied to low-level problem. As inputs, the model takes the species-trait matrix Q and the task-trait matrix Y derived from the clusters obtained from the expert demonstrations. It outputs the assignment matrix X and the strategy Z.

Each species has several traits or capabilities. These are encoded by a species-trait matrix, $Q = \{q_1, q_2, \dots, q_S\} \in \mathbb{R}_+^{S \times U}$, where $q_s \in \mathbb{R}_+^U$ is the trait vector listing the different traits of the s^{th} species and U is the total number of traits. The assignment of robots to tasks is encoded by the assignment matrix $X = [x_{ms}] \in \mathbb{Z}_+^{M \times S}$, where the ms^{th} element x_{ms} denotes the number of robots from species s assigned to task T_m. The aggregated traits available at each task, at the time of assignment, is encoded in the task-trait matrix $Y \in \mathbb{R}_+^{M \times U}$, whose mth row represents the aggregation of traits available for task T_m and is given by

$$y_m = Q^T x_m, \forall_m \in \{1, \dots, M\}. \quad (1)$$

For each mth task there is a set of P_m strategies. Each strategy denotes the traits required to complete the task. As such, the r^{th} strategy for the m^{th} task is given by

$$^r y_m^* \in \mathbb{R}_+^U, \forall r \in \{1, \dots, P_m\}, \quad (2)$$

where P_m represents the number of strategies for task T_m. There can be more than one set of strategies that result in successful completion of the same task, and it is assumed that satisfying any one of them will suffice to achieve completion of that task.

For training purposes, the model is presented with a set of N ***expert demonstrations*** of robot assignments. These are given by

$$D = \{X^{(i)}, Q^{(i)}\}_{i=1}^{N}, \tag{3}$$

where $X^{(i)}$ represents the expert-specified assignment matrix for a team with capabilities encoded by the species-trait matrix $Q^{(i)}$.

Given these definitions, the GT-RSA algorithm has distinct training and testing modes. In training mode, strategies for each task are extracted from expert demonstrations D (Fig. 2). As inputs, the model takes optimal task assignments X and game states (species-trait matrix) Q and computes the task-trait matrix Y. The rows of Y represent the aggregated traits for each task. Clusters of aggregated traits in Y are identified as optimal strategies represented by their centroids. Clustering is performed independently for each task on each row Y_m (rows Y_m are task subspaces of Y).

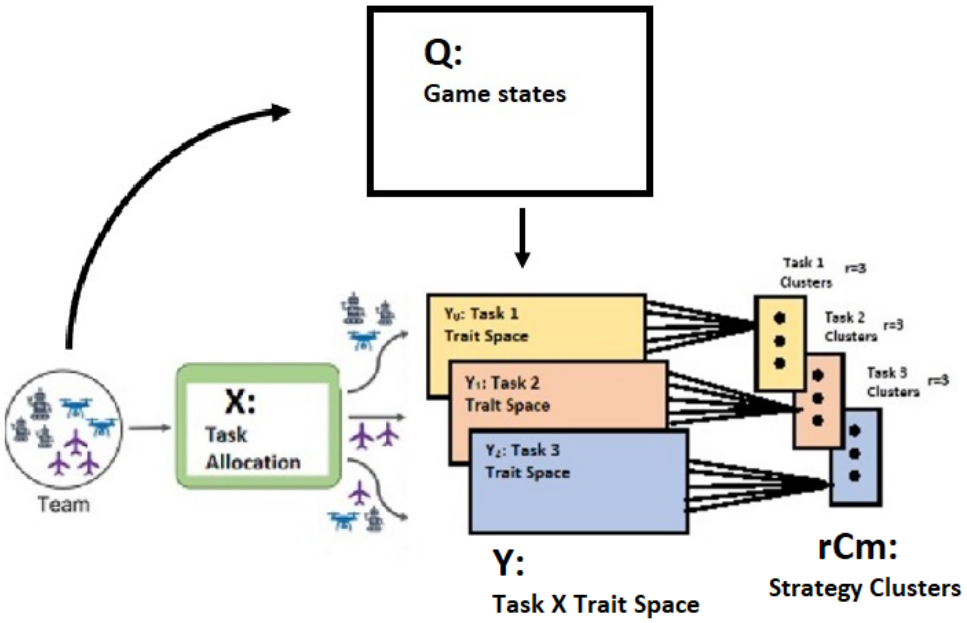

Fig. 2 Training mode, where similar aggregated traits (Y = QX) are grouped into strategy clusters $^{r}C_m$

In testing mode, the model uses these strategies to generate an assignment matrix $X^{(j)}$ when provided with partial inputs (Q, N_s).

2.1.1 Model Training Phase

In training mode, the GT-RSA algorithm performs unsupervised clustering of expert demonstrations and outputs their centroids. Training data consists of expert demonstrations (Eq. 3). Clustering is performed on aggregated traits (Y) computed using Eq. 1. In a modular fashion, each row of Y is independently analyzed by an unsupervised clustering algorithm that monitors variance to partition the aggregated traits of that row into P clusters:

$$\{ {}^{r}C_m \}_{r=1}^{P_m}, \tag{4}$$

where P_m represents the total number of clusters for the row (representing task m). ${}^{r}C_m$ is an index running over the clusters for each task m, double indexed by cluster number r (r ranges between 2 and 5 depending on the row). Cluster ${}^{r}C_m$ is defined by its centroid, a vector of aggregated traits ${}^{r}\hat{y}_m$:

$${}^{r}\hat{y}_m = \sum_{y_m^{(i)} \in {}^{r}C_m} \frac{y_m^{(i)}}{|{}^{r}C_m|}, \tag{5}$$

where $|{}^{r}C_m|$ denotes the number of datapoints in the r^{th} cluster of task m. The number of clusters was fixed at three in FY22 and Srikanthan and Ravichandar (2022). FY24 used a variable number of clusters depending on an optimization algorithm that used four statistics to select the number of clusters: 1) within cluster sum of squares, 2) silhouette scores, 3) relative entropy (which increases when clusters have balanced counts), and 4) a penalty based on the number of singleton clusters. This algorithm produced a range of 2–5 clusters.

2.1.2 Model Testing Phase

In testing mode, partial inputs (Q, N_s) are matched to the nearest centroid and an assignment matrix is output by solving the inverse problem to Eq. 1. Test data consists of the maximal number of species available for allocations and a game state: <N_s, Q>. Aggregated traits are computed for these inputs (Eq. 1) and the values for each task are compared against cluster centroids (Eq. 5). The best available matching cluster is chosen as the strategy for that task. To determine the nearest cluster, a trait mismatch error is computed for each task

$${}^{r}e_m = \left\| {}^{r}\hat{y}_m - Q^{(j)^T} x_m \right\|_2^2, \tag{6}$$

where x_m reflects N_s input values. Hence the model acts as a winner-take-all classifier for each task, matching a novel input to learned centroids for each task. The winner-take-all output from the clusterer is encoded in a one-hot vector of m strategy selectors $z_m \in \{0,1\}^{Pm}, \forall m.$

While Eq. 6 determines which clusters are nearest, the winner must also satisfy several constraints: 1) the assignment matrix $X^{(j)}$ must be possible given N_s, 2) only one cluster can be picked per row, and 3) the trait mismatch error must be negative:

$$y_m \succ {}^{r}\hat{y}_m, \forall_m \,, \tag{7}$$

where y_m denotes the traits aggregated by the target team towards the m^{th} task and $\succ$ denotes the element-wise greater-than operator. When considered jointly with other constraints, this help ensures y_m stays close without going under the value of ${}^{r}\hat{y}_m$.

2.2 FY22: Application of GT-RSA Model to Low-Level Problem

The aim of this portion of the study is to evaluate the application of the GT-RSA algorithm to the low-level mounted platoon problem by assigning crew stations to vehicle subsystems. Because crew members can operate any of the team's subsystems from their crew station, we are going to assume that crew station and crew member are synonymous. Here the problem is flipped around somewhat, as we are assigning subsystems to crew stations, rather than the reverse. Therefore, the various vehicle subsystems comprise the robot species.

The problem formulation for the data from the Information for Mixed Squads (INFORMS) FY22 data is as follows:

- A total of six armored ground vehicles (MCV1, RCV2, RCV3, MCV4, RCV5, and RCV6), where two are manned (MCVs) and four are unmanned/robotic (RCVs).
- Each armored ground vehicle has four subsystems (mobility, lethality, slewable sensor, UAV); each subsystem can only be controlled by one crew member at a time.
- A total of 14 crew members (CS01–CS14); CS01–CS07 can control any of the subsystems on MCV1–RCV3 and CS08–CS14 can control any of the subsystems on MCV4–RCV6.

A mounted platoon consists of two teams of three vehicles. Each team consists of one MCV and two RCVs. Each vehicle has four unique subsystems: lethality, mobility, slewable sensor, and UAV (Table1). Each team has seven crew members assigned to each of the seven crew stations in the MCVs. So, the question becomes how to assign these 24 vehicle subsystems to 14 crew stations?

2.2.1 Methods

Vehicle substations are enumerated in Table 1 and the traits of the vehicle subsystems vary and are defined in Table 2.

Table 1 Species and vehicle subsystems for low-level problem

Vehicle species and subsystems		
MCV1-Lethality	9. RCV3-Lethality	17. RCV5-Lethality
MCV1-Mobility	10. RCV3-Mobility	18. RCV5-Mobility
MCV1-Slewable	11. RCV3-Slewable	19. RCV5-Slewable
MCV1-UAV	12. RCV3-UAV	20. RCV5-UAV
RCV2-Lethality	13. MCV4-Lethality	21. RCV6-Lethality
RCV2-Mobility	14. MCV4-Mobility	22. RCV6-Mobility
RCV2-Slewable	15. MCV4-Slewable	23. RCV6-Slewable
RCV2-UAV	16. MCV4-UAV	24. RCV6-UAV

Table 2 Traits for the vehicle subsystems

Traits
Health: the health value (0–100) of the subsystem (should be 100 for UAV always)
IsFiring: 1 if the vehicle the subsystem is part of has recently fired, and 0 else
IsMoving: 1 if the vehicle the subsystem is part of has recently moved, and 0 else
IsDamaged: 1 if the vehicle the subsystem is part of has recently been hit, and 0 else
IsConnected: 1 if the vehicle has non-zero telemetry link, and 0 else (should be 1 for MCVs always)
IsSlewable: 1 for lethality, slewable, and UAV, and 0 else (is static for all data points)
IsWeapon: 1 for lethality, and 0 else (is static for all data points)
IsUsableOnMove: 1 for mobility, lethality, and slewable, 0 for UAV (is static for all data points, this is whether the subsystem is usable on the move)
IsManned: 1 for MCVs, 0 else (is static for all data points)
Section: 0 for MCV1, RCV2, RCV3,1 for MCV4, RCV5, RCV6 (is static for all data points)
Elevation: 0 for mobility, 1 for lethality and slewable, 3 for UAV (is static for all data points)

2.2.1.1 Strategies

Multiple strategies can be used to meet task demands, so these are extracted with clustering in the trait space. As per the GT researchers, the trait aggregations are computed from the demonstrations D using Eq. 1. Trait aggregations are likely to form clusters because there are likely to be multiple strategies that will solve each task. That is, it is less important which vehicle substations come into play and more important which clusters of traits, denoted by $\{ {}^{r}C_m \}_{r=1}^{P_m}$, are present.

We arbitrarily chose to identify three clusters per task, following the example of the GT researchers, but this is not critical. Once the clusters were identified, the task requirements were computed from the strategies required for each task using Eq. 5 (task requirements are the aggregated traits necessary to complete a task).

To solve our problem, we need to both choose the strategies (trait aggregations) required for each of our M tasks and optimize allocation of available resources (traits) to all the tasks. This is done using strategy selectors $z_m \in \{0,1\}^{Pm}, \forall m$, which are one-hot encoded decision variables that identify the choice of strategy for each task. To optimize the allocation of strategies to tasks, we use integer decision variables $x_m \in \mathbb{Z}_{+}^{S}$, $\forall$m representing the assignment of robots to each task. The degree to which all tasks get allocated sufficient resources is measured in terms of the degree of mismatch. So, for each task T_m that gets assigned the r^{th} strategy, the corresponding trait mismatch error is defined in Eq. 6.

The goal for a novel team of vehicle subsystems and set of tasks is to minimize the sum of the trait mismatch error across all tasks, as expressed here:

$$E_m = z_m^T e_m \,, \tag{8}$$

where $e_m = \left[{}^{1}e_m, \ldots, {}^{PM}e_m \right]^T \in \mathbb{R}_{+}^{Pm}$ is a vector of trait mismatch errors between the aggregated traits and each of the strategies' task requirements.

2.2.1.2 Algorithm

The ***resource aware*** optimization of assignments of vehicle subsystems to crew stations is defined as a constrained quadratic integer program:

$$\left\{x_m^{*(j)}, z_m^{*(j)}\right\}_{m=1}^{M} = \; arg^{min}_{x_m z_m} \textstyle\sum_m E_m \,. \tag{9}$$

The GT researchers add the constraint that the number of assignments is less than or equal to number of robots per species. However, as defined here, there is only one vehicle subsystem per species (this could arguably have been structured otherwise) so this constraint is unnecessary. Furthermore, as there is only one of each species, they are either assigned to a task or not (z_m is binary). Finally, there is the constraint that the total of the resources or traits allocated to the tasks cannot exceed the number of resources available.

In our application, the vehicle subsystems 1–12 can only be assigned to crew stations 1–7, and 13–24 to crew stations 8–14; thus, this was added as a constraint to the model.

The data from the FY22 INFORMS study was used for expert demonstrations. Ward's Euclidean distance clustering was used to develop clusters of required traits for the task-trait matrix. The GT researchers used 3 clusters, so this is what was used here.

As proof of concept, we turned the GT algorithm into an incremental update algorithm using iterative error reduction. Assignment (X') and strategy (Z') variables were randomly initialized. Next, the trait-mismatch error was computed (Eq. 8) and (X') and (Z') were updated to reduce the error. This process was repeated for 100 randomly initialized assignment and strategy variables. Figure 3 demonstrates how the dataset was created using an iterative (repeated error update) optimization method. 100 random inputs $\{X', Z'\}$ were replaced by the values generated after 15 error-based updates. The resulting dataset should closely resemble the strategy clusters learned from the INFORMS data. Figure 4 shows the distribution of error after the iterative (repeated error update) optimization method.

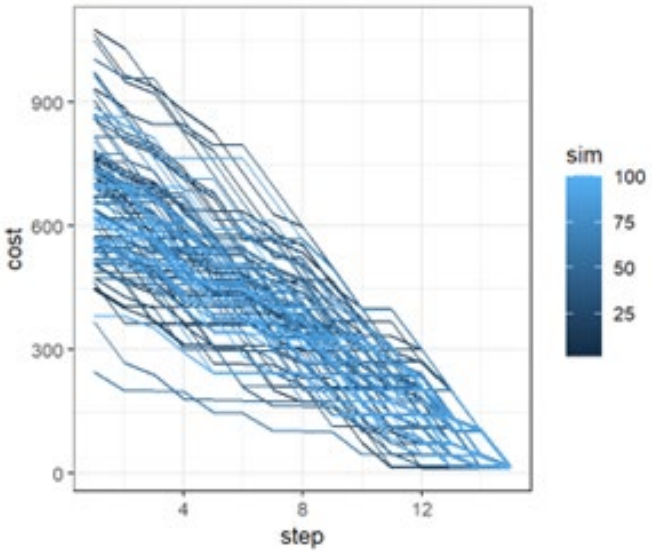

Fig. 3 Trait-mismatch error (cost) as measured for 100 randomly initialized simulations

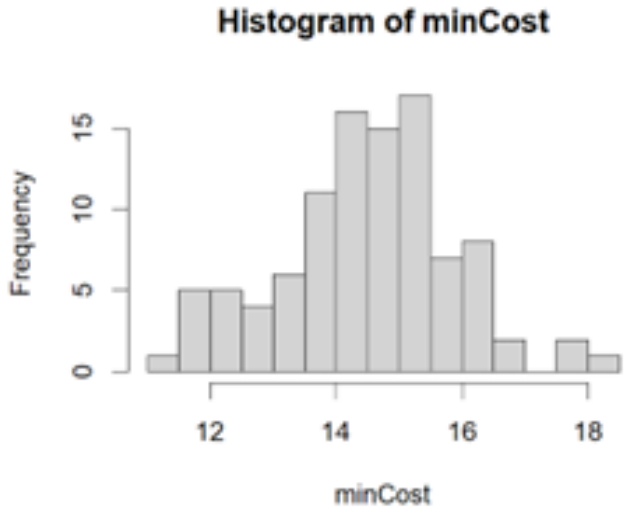

Fig. 4 Minimum trait-mismatch error (cost) for 100 data points after optimization

The minimum cost ranged from approximately 11 to 18 (Fig. 4). Note that this process does not guarantee that the algorithm has converge or that this is the absolute minimum, although the cost is monotonically decreasing across steps (as shown in Fig. 2).

2.2.2 Results and Discussion

One way in which the performance of this algorithm can be assessed is by comparing the match between the inferred assignment matrix X' and the observed assignment matrix X on held-out data. To do this, a random 100 instances were selected as hold outs (i.e., a test set). With the remaining data (the training set), the clustering algorithm was applied to obtain sets with four possible numbers of clusters, $P^m = \{2,3,4,5\}$. Note that number of strategies remained fixed per task. For each possible number of clusters, the task requirements (per Eq. 5) were calculated from the training data. Then, for each of the held-out instances, for each of the possible cluster amounts, the optimization method described above was used to extract the best inferred assignment matrix, X', and strategy selection matrix. Twenty simulations were generated for each instance and the result that produced the lowest cost was selected.

The inferred assignment matrices were then compared with the observed assignment matrices associated with each held-out instance in terms of percent match: each element of X' was given a 1 if it matched the same element in the original X, or a 0 otherwise. The mean was then taken of these binary values. Percent match for each held out instance for each assumed number of clusters is plotted below in Fig. 5. The red lines in Fig. 5 represent the range of match observed for 20 randomly initialized matrices for that instance. The black lines represent the match after optimization.

As a means of judging the performance of the algorithm against chance, the randomly generated X' used to initialize each of the 20 simulations for each instance and crew station were also compared with the original X. This distribution of percent match values for each simulation forms a kind of surrogate confidence interval (CI) that gives a sense of what performance would look like under random assignment and strategy selection. In Fig. 5, the red bands show the upper and lower limits of this 95% CI. It is clearly the case that the algorithm outperforms chance in most situations when there are two, three, or four clusters. However, assuming five clusters leads to a clear loss in performance.

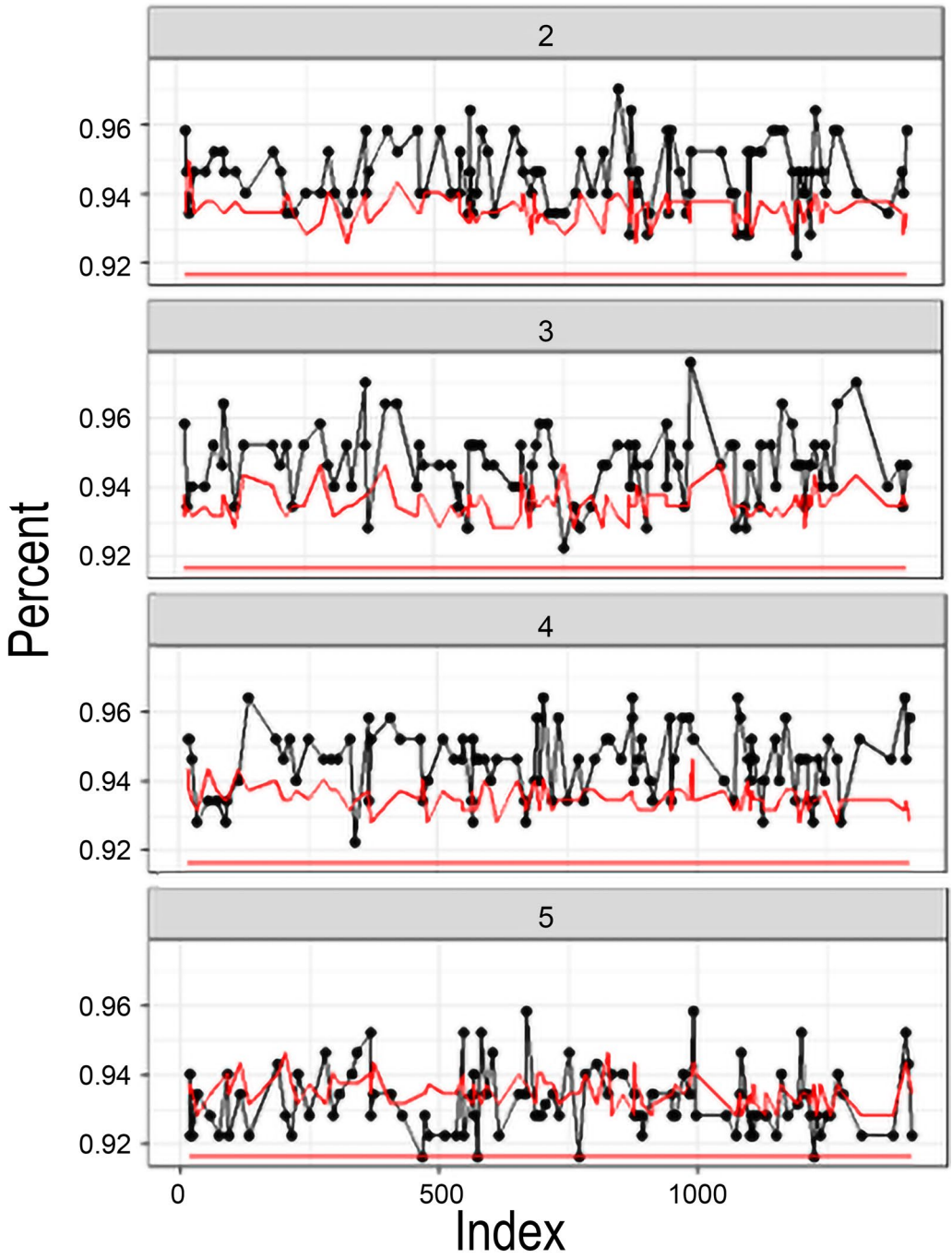

Fig. 5 Percent match between the inferred optimized matrices and the observed assignment matrices for each of 100 hold-out instances

2.3 FY24: Application of GT-RSA Model to High-Level Problem

The aim of this portion of the study is to evaluate the application of the GT-RSA algorithm to the high-level mounted platoon problem by comparing its predictions when it is trained on two distinct sets of expert-generated demonstration data: one

with good strategies another with bad strategies. The inclusion of both static and dynamic traits is novel, and the modification was made to make tasking context sensitive.

Application of the GT-RSA algorithm to the high-level problem requires identifying a set of tasks (T_m), a team of vehicle species (S, N_s) and a set of vehicle traits (U).

- The team consists of six armored ground vehicles (MCV1, RCV2, RCV3, MCV4, RCV5, and RCV6), where two are manned (MCVs) and four are unmanned/robotic (RCVs) along with six unmanned aerial drones (UAV1, UAV2, UAV3, UAV4, UAV5, and UAV6). Each vehicle is either assigned (1) or unassigned (0) to a given task. Vehicles are listed in Table 3 and tasks in Table 4.
- Each vehicle has four static traits that define the specializations of each vehicle (manned, aerial, armor, armament) and 15 dynamic traits that vary with game history. These traits defined in Table 5.

Table 3 Vehicle species (UAVs are linked to host vehicles with same number)

Vehicle species			
MCV1	MCV4	UAV1	UAV4
RCV2	RCV5	UAV2	UAV5
RCV3	RCV6	UAV3	UAV6

Table 4 Tasks available for missions

Tasks
Phase line: move and assemble at the border to the next battle map
Battle position: assemble and move in a convoy
Key terrain: move and occupy a secondary target location
Objective: move and occupy a primary target location
Named AOI: move and perform reconnaissance at an area of interest
Attack: counter an attacking hostile
Support: provide ranged support to a forward vehicle
Present hostile: move to and attack confirmed hostile
Anticipated hostile: move and confirm presence of anticipated hostile

Note: AOI = area of interest

A central question was whether the GT-RSA model would make realistic predictions when trained on realistic training data. In the first study two sets of expert generated demonstration data were used to train the models. One dataset illustrated good strategies and the other bad. Simulations reveal the model fails to make distinct predictions for these datasets. Analysis of this failure indicated that the demonstration data likely violated some model assumptions regarding time, so a new dataset was extracted from the original and a second set of experiments

performed. For the second experiment it was hypothesized that distinct GT-RSA models trained on distinct good and bad strategies will make different predictions when presented with the same test dataset. This was found to be true, but the model was also observed to generate highly variable outputs in general.

Table 5 Static and dynamic vehicle traits expressed on a 0–100 scale

Traits
Manned: vehicle specific static trait; 100 if the vehicle is manned 0 otherwise
Aerial: vehicle specific static trait; 100 if the vehicle is aerial 0 otherwise
Armor: vehicle specific static trait; ordinal scaling (heavy or 100, 50 or light, 0 or no armor)
Armament: vehicle specific static trait; ordinal scaling (heavy or 100, 50 or MG only, 0 or no armament)
Mobility: the health value (0–100) of the mobility subsystem
Primary: the health value (0–100) of the primary subsystem (usually weapons)
Auxiliary: the health value (0–100) of the auxiliary subsystem (usually sensors)
Link: the health value (0–100) of the MCV–RCV link
Antenna: the health value (0–100) of the UAV link on the host vehicle
UAV battery: the charge of the UAV battery (0–100), represented on UAV and host vehicle
UAV: the health of the UAV (0–100), represented on UAV and host vehicle
AP: the inventory of armor piercing ammunition as percent of max value (0–100)
HE: the inventory of high explosive ammunition as percent of max value (0–100)
MG: the inventory of machine gun ammunition as percent of max value (0–100)
COAX: the inventory of coaxial MG ammunition as percent of max value (0–100)
Moving: binary state of vehicle indicating stationary or moving (0, 100)
Detecting: binary state of vehicle indicating sensing or not (0, 100)
Engaged: binary state of vehicle indicating enemy attack (0, 100)
Engaging: binary state of vehicle indicating in active exchange/conflict (0, 100)

2.3.1 Methods, Assumptions, and Procedures

FY24 algorithm was implemented using Python, pkl, and csv data files. Code and data are maintained in a sitcore repository.* Model parameters are set out in config.py. Results are displayed in a jupyter report GT_RSA_REPORT.ipynb, which links to images in a ./img subdirectory and data classes in INFORMS_pilot_classes.py. Source data for experiment 1 is found in a demodata subdirectory with the files CJMD_allocations16b.csv and CJMD_examples16_handmade.csv. Data for experiment 2 is derived from experiment 1 in the jupyter report. Past simulations can be loaded from the last run on 9 September 2024 and consists of a series of csv and pkl files of form Good_*_090924.csv or Bad_*_090924.csv.

2.3.2 Experiment 1: Demonstration Data

Demonstration data was created and encoded into the variables described by the GT formalism. One scenario was created by an expert in mounted platoon tactics and a second scenario was created based on the underlying principles. Figure 6

* https://gitlab.sitcore.net/arl/hred/hat-erp/project-6/gatech-arl-ca

illustrates examples from each scenario. Figure 7 shows examples where these scenarios were coded into assignment matrices (X) and game state matrices (Q) using the variables defined in Tables 3–5.

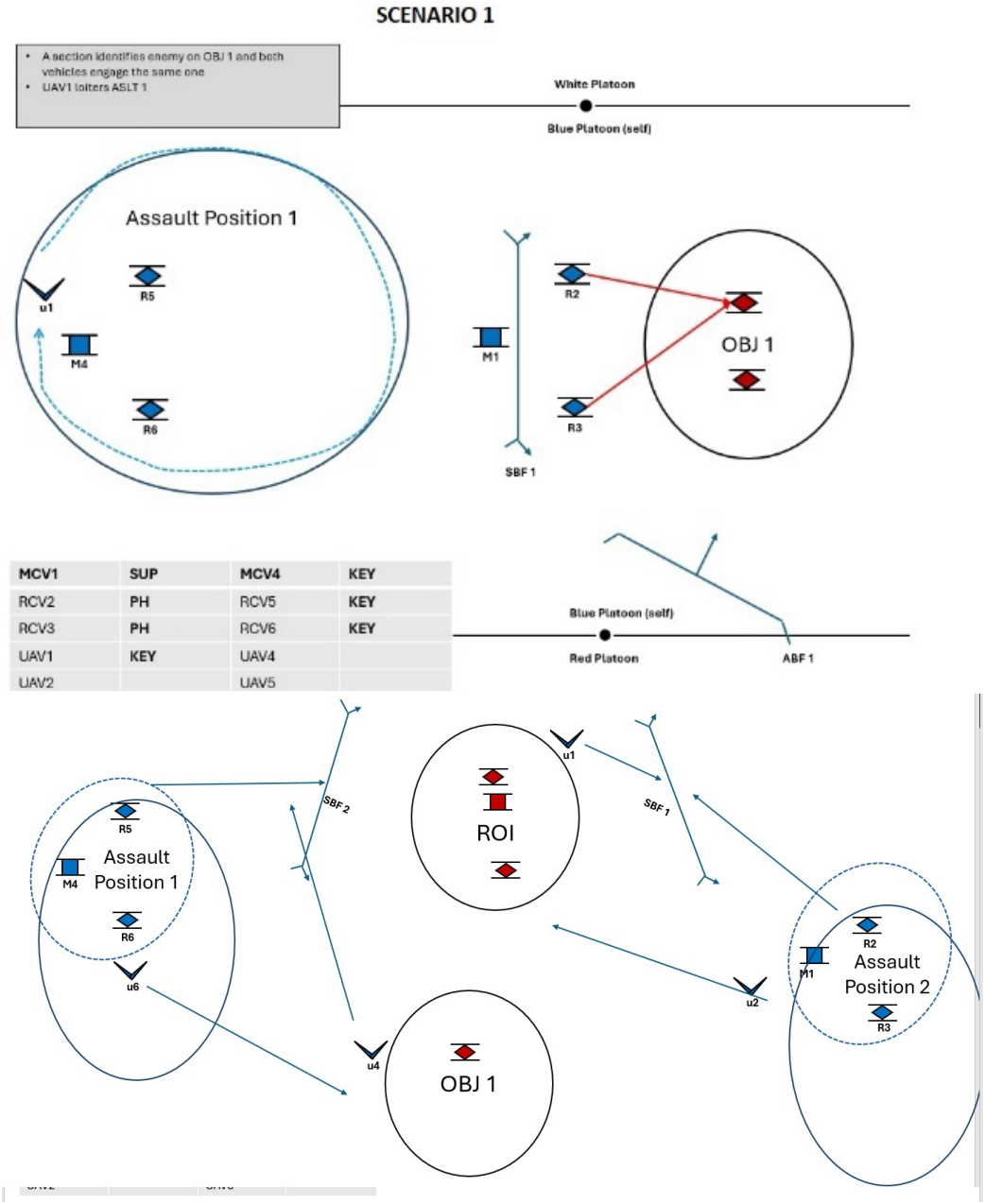

Fig. 6 Examples of demonstration data battle scenarios. (Top) Scenario generated by expert on mounted platoon tactics including flanking operations, surveillance, and force support. (Bottom) Second scenario elaborating on these tactics.

Q

Agent	UAV7	UAV8	UAV9	UAV10	UAV11	UAV12				MCV1	...					
State	N	N	N	N	N	N	Manned	Aerial	Armor	Armament	...	Antenna	Battery	AP	HE	MG
0	1	1	1	1	1	1	100	0	100	100	...	100	100	0	0	0
1	1	1	1	1	1	1	100	0	100	100	...	100	100	0	0	0
2	1	1	1	1	1	1	100	0	100	100	...	100	100	0	0	0
3	1	1	1	1	1	1	100	0	100	100	...	100	100	0	0	0
4	1	1	1	1	1	1	100	0	100	100	...	100	100	0	0	0

X

Agent	MCV1									
Task	Support	Present_Hostile	Anticipated_Hostile	Phase_Line	Battle_Position	Key_Terrain	Objective	Named_AOI	Attack	S
0	0	0	0	0	0	1	0	0	0	
1	0	0	0	0	1	0	0	0	0	
2	1	0	0	0	1	0	0	0	0	
3	1	0	0	0	0	0	0	0	0	
4	0	0	0	0	1	0	0	0	0	

Fig. 7 Samples of game state/trait matrix Q and assignment matrix X. Two tables representing matrices N_s, Q and X. Each row indexes a game. (Q) first six values are from N_s and show the count of UAVs; remaining values are linearized game state values for the matrix Q (score range is [0–100]). (Bottom) linearized X matrix with task assignments for each vehicle. X is binary. Variables as in Tables 3–5.

Trait aggregations were computed per Eq. 1 and analysis of variance (ANOVA) and t-tests were run comparing trait aggregations for good and bad strategies on each task (i.e., $y_{m,\,good}$ vs. $y_{m,\,bad}$). Silhouette scores were also computed comparing good and bad strategies on each task. A performance metric from the paper, percent of vehicles deployed (agent utilization) compared good and bad strategies using a t-test.

2.3.3 Experiment 2: Model Performance on Extracted Stochastic Data

2.3.3.1 Agglomerative Clustering Methods

The aggregated traits associated with each task were analyzed using agglomerative clustering (sklearn toolkit, Ward's linkage, Euclidean metric, number of clusters ranging from 2 to 10). The number of clusters identified for each task was determined using the methods for FY24 as described in Section 2.2.1.1. The number of clusters varied between 2 and 4 for each task.

2.3.3.2 Synthetic Data Creation

A new dataset was created for the second experiment using the good and bad strategy clusters identified in demo data from the first experiment. Good and bad synthetic data was created by randomly selecting a centroid for each task. With nine tasks and an average of three clusters per task, approximately 3^9 or approximately $20{,}000$ synthetic strategies are possible. The aim is to find good and bad strategies

that are the most different, so silhouette scores were calculated for 70,000 pairs of good and bad strategies randomly selected from the space of approximately $20{,}000^2$ possible pairs of good and bad strategies. The top 32 pairs were picked to form the basis of a new dataset. ANOVA, silhouette scores, and t-tests verified this new synthetic dataset was distinct and linearly separable. The demo data associated with these 32 pairs was retrieved and used to compute a mean game state matrix (Q) for synthetic cluster combination. Since Q is not indexed by task, it represented the mean Q over all data that was incorporated into a given cluster combination. Figure 8 shows two typical results from this approach demonstrating it generated separable good and bad strategies. Red points chart the computed silhouette scores comparing n = 16 good strategies and n = 16 bad strategies for each task. The blue box and whisker plots indicate the range and mean silhouette score from 500 random labelings of the same data.

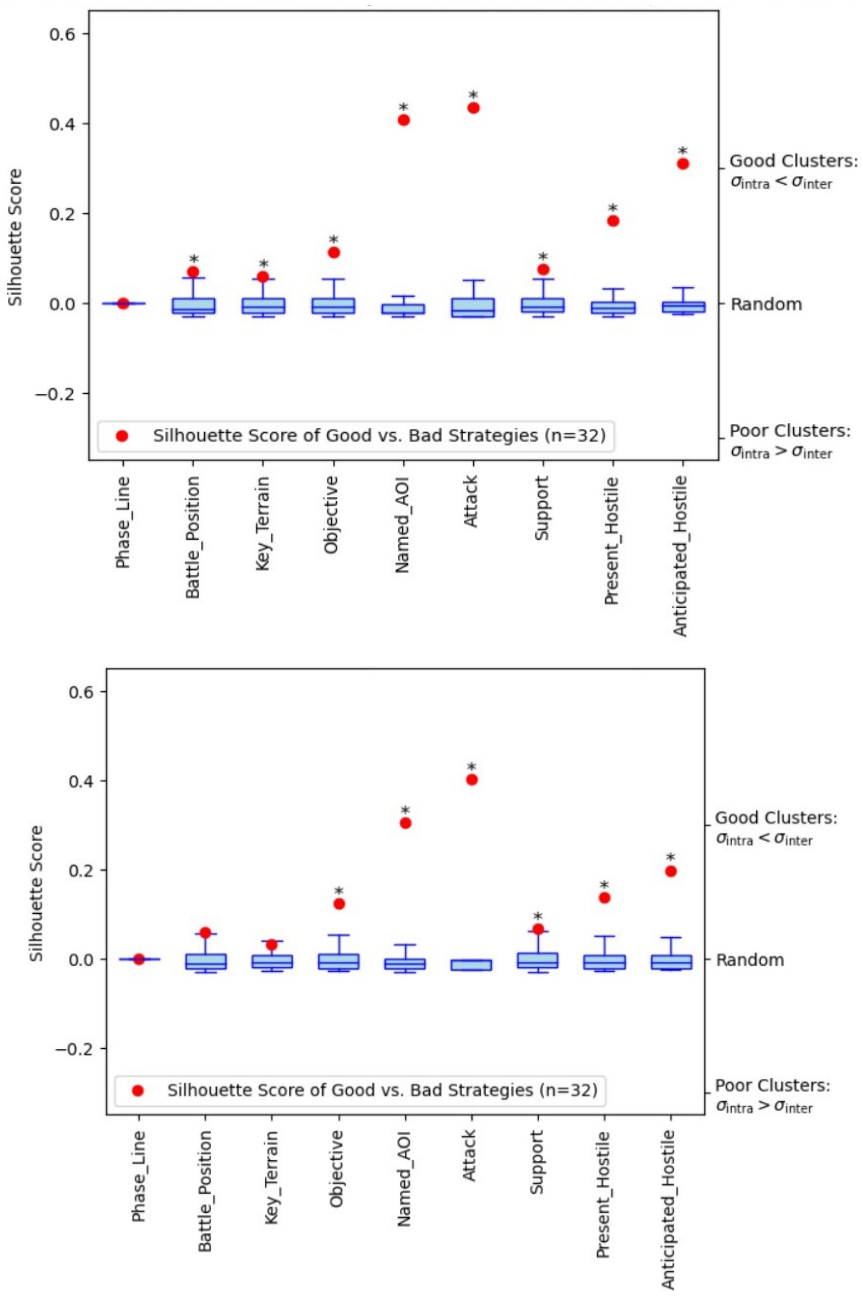

Fig. 8 Silhouette scores for trait aggregations (Y_m) synthesized from good and bad data clusters (95% CI, random 500 labels). (Top and Bottom) Silhouette scores from two randomly generated sets of synthetic data.

2.3.3.3 Stochastic Data Creation

The new training dataset was created by using the synthetic centroids as seeds in a multivariate Gaussian. This was done to generate new aggregated traits Y and game states Q. A binary vector representing N_s was generated and X was generated by minimizing $Y - XQ$ under constraints of N_s. This almost always converged to a solution because there were fewer constraints than the GT-RSA optimization

problem that simultaneously selects winning strategies. ANOVA, silhouette scores, and t-tests verified this new stochastic dataset was distinct and linearly separable. The stochastic data was separated into training ($n = 1000$) and testing datasets ($n = 200$) for each strategy case. Figure 9 shows two typical results from this approach demonstrating it generated separable good and bad strategies. Red points chart the computed silhouette scores comparing $n = 1000$ good strategies and $n = 1000$ bad strategies for each task. The blue box and whisker plots indicate the range and mean silhouette score from 500 random labelings of the same data. Top and bottom figures correspond to datasets stochastically generated from seeds in top and bottom examples from Fig. 8.

Green triangles represent an application of FY22 methods to the stochastically generated data points (Fig. 9); these represent mean silhouette scores for data after iterative correction towards clusters.

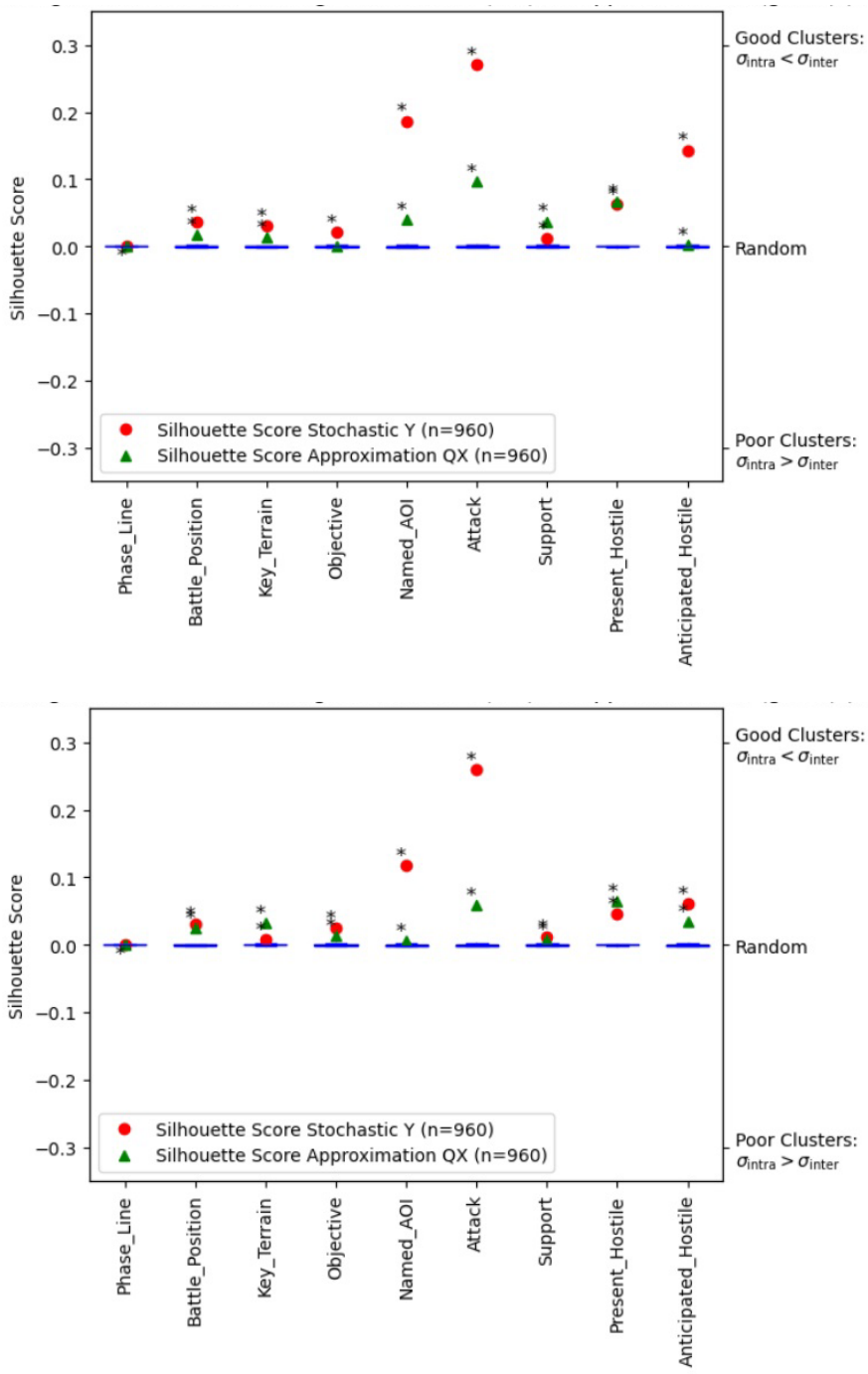

Fig. 9 Silhouette scores stochastically generated from good and bad synthetic trait aggregations (Y_m). Exact Y (red) vs. approximate XZ (green) (random labels in blue).

2.3.3.4 Model Training and Testing

Two GT-RSA models were trained separately on good ($n = 1000$) and bad ($n = 1000$) stochastic datasets and then tested on 100 hold outs from each test set ($n = 200$). Training data consisted of $<X, Q>$ pairs representing either good or bad

strategies. Training established the (aggregated trait) centroids for each task. For testing, each model is presented with <N_s, Q> pairs from the hold-out test set ($n = 200$; 100 from good hold out, 100 from bad hold out). Model outputs for the shared test set were compared using Hamming distance, silhouette scores, Jaccard similarity, agent utilization, and failed solution counts.

2.3.4 Results and Discussion

2.3.4.1 Experiment 1 Results

Experiment 1 aimed to test whether the model could discriminate between good and bad expert-generated demonstration data. This is dependent on the signals being different. Figure 10 plots the mean aggregated trait values for each task and indicates there were no detectable differences between signals in any task. The mean aggregated traits y_m were computed for each task T_m. Good strategies are blue ($n = 16$) and bad are red ($n = 16$). The standard error is plotted alongside both as a shaded interval. No comparisons were significant using permutational multivariate ANOVA. The attack task has an $n = 2$. Axes are variable. Figure 11 plots the silhouette score, which is a metric comparing intraclass variation and interclass variation. Red line charts the computed silhouette scores comparing $n = 16$ good strategies and $n = 16$ bad strategies for each task. The blue line plots the mean silhouette score from 300 random labelings of the same data; the shaded blue line indicates the 95% CI corrected for multiple comparisons. No silhouette scores are significantly different than chance. Interclass and intraclass variation were similar in magnitude between good and bad strategies indicating they had no natural clustering above chance.

Fig. 10 Mean aggregated traits of good and bad strategies in demonstration data do not differ

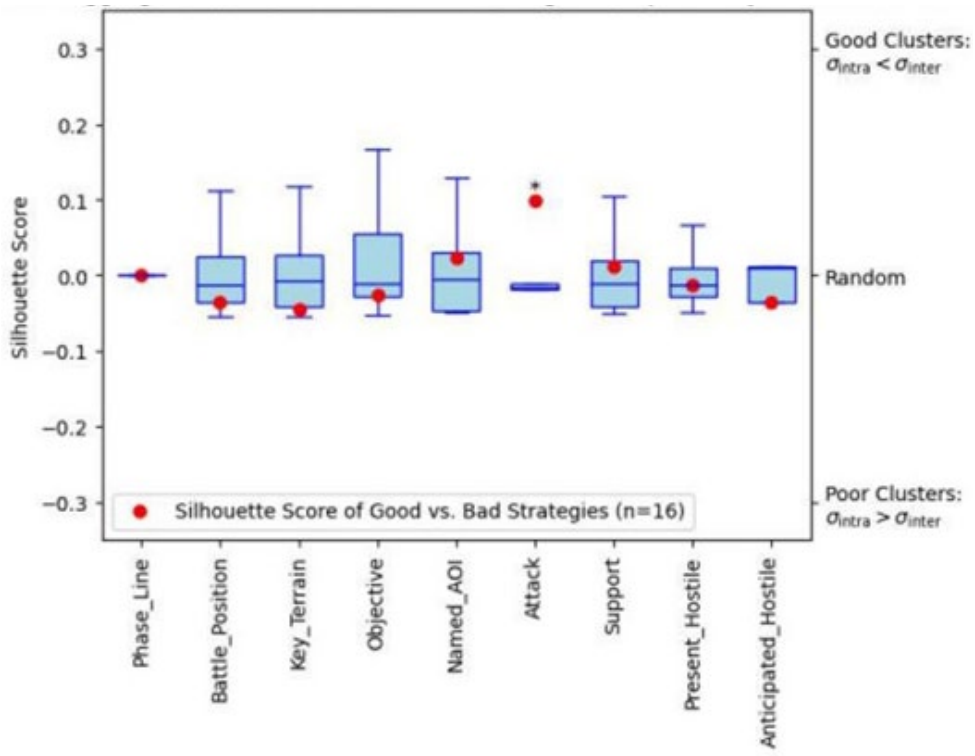

Fig. 11 Silhouette scores between good and bad strategies of demonstration data do not differ

Figure 12 indicates that good and bad strategies did differ in terms of vehicle utilization. Good strategies had a significantly higher agent utilization. Significant

differences were observed between good and bad strategies (t-test). Units are percent of units tasked in each scenario <X, Q>.

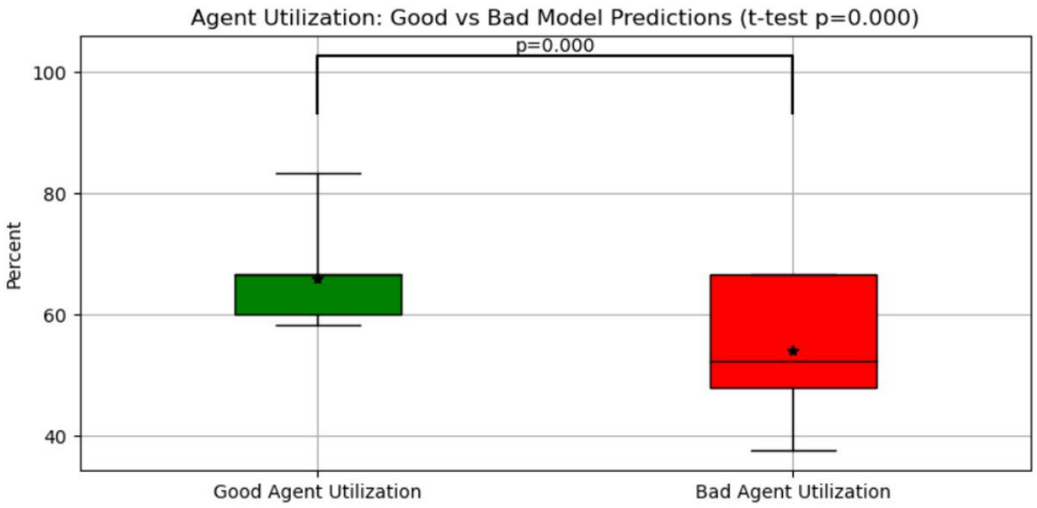

Fig. 12 Agent utilization differs between good and bad strategies in demonstration data

2.3.4.2 Experiment 2 Results

The second experiment explored whether models trained on distinct stochastically generated datasets would make different predictions on a shared test dataset. A new dataset of distinct good and bad strategies was created from the demonstration data using the approach as described in section 2.3.3.2. Figure 13 shows that the mean aggregated traits from good and bad datasets are statistically distinct in most tasks.

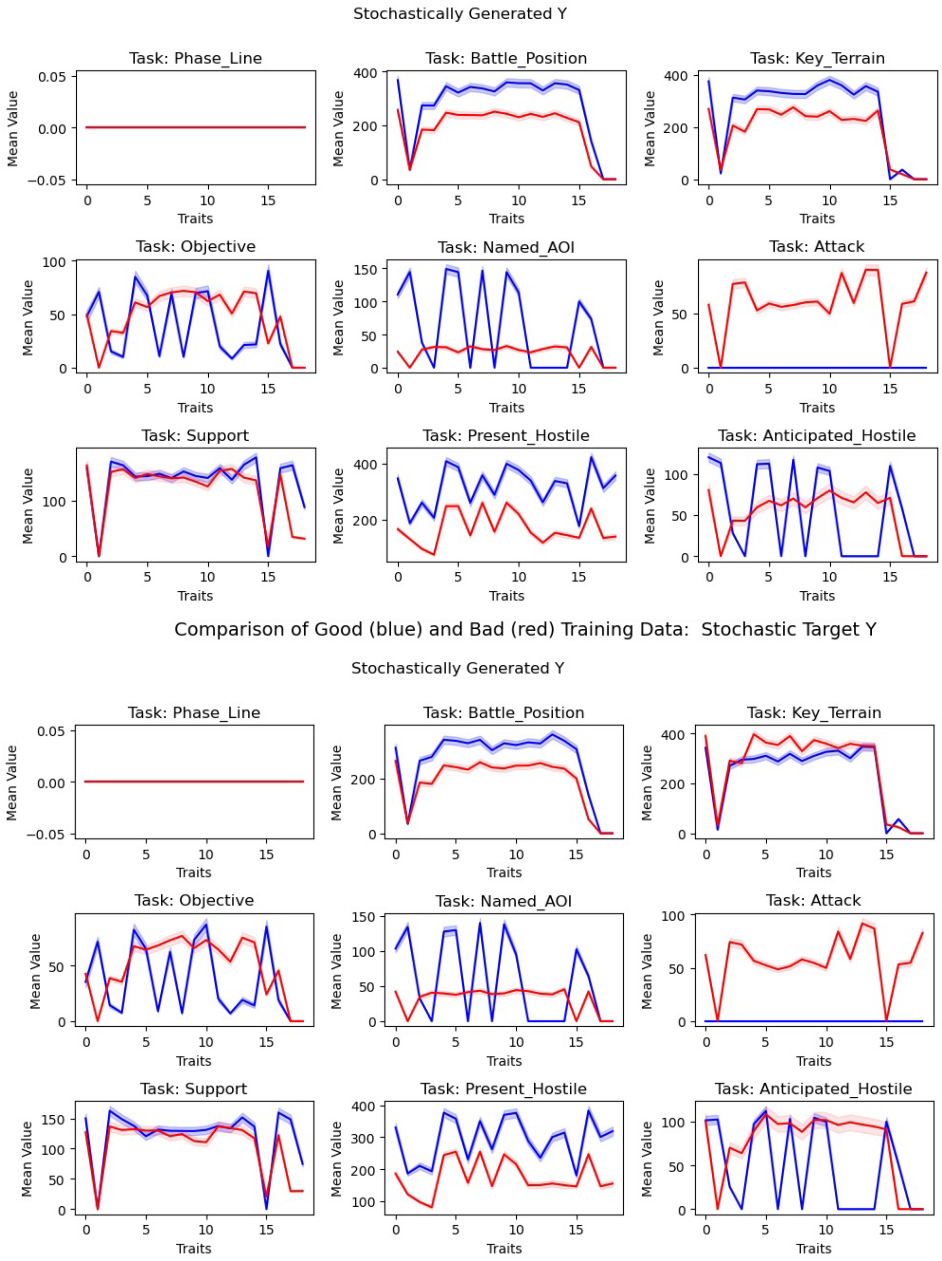

Fig. 13 Aggregated traits for stochastically generated good and bad strategies differ significantly. The mean aggregated traits Y_m were computed for each task T_m. Good strategies are blue (n = 1000) and bad are red (n = 1000). The standard deviation is plotted alongside both as a shaded interval. (Top and Bottom) Different simulations that correspond to top and bottom simulations in Figs. 8 and 9.

Models trained on different datasets were tested on a common dataset. The results from two different experiments are shown in Fig. 14, with (14, left) and without (14, right) null allocations for tasks (a null allocation means the trivial solution of all zeros was returned by optimizer). The left side of Fig. 14 shows around 25% of model predictions were null (red bars; optimization was done using the library docplex). The right side of Fig. 14 shows that if these null predictions are excluded,

the models never made the same allocation matrix prediction (purple). Blue bars indicate that around 60% of task allocations are the same when compared row by row. The average difference between rows that are not the same is about 0.1 bits, meaning 1 to 2 values differ per task.

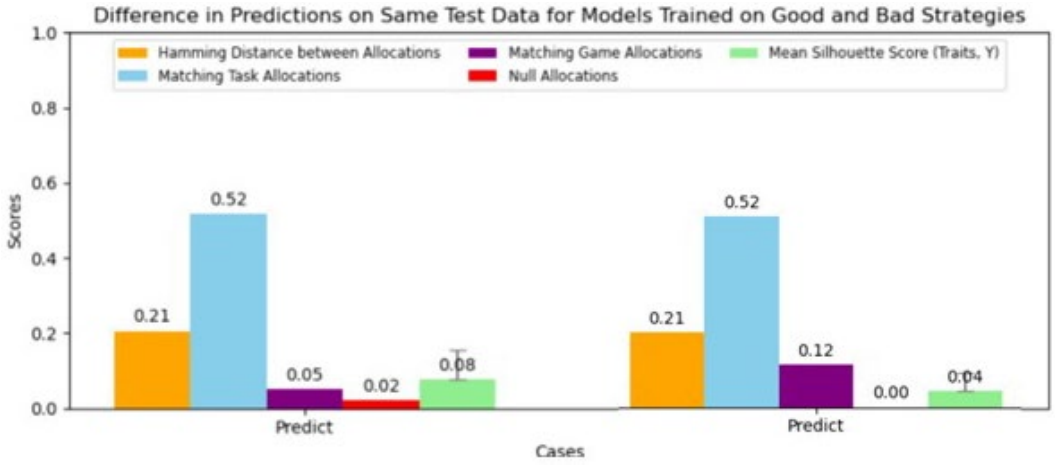

Fig. 14 Models trained on different datasets make different predictions when given the same inputs. Bar plots from 2 simulations (bagplots on left and right, corresponding to left and right cases in Figs. 8, 9, and 13). Left, simulation comparing predictions from two different models on the same dataset; one model is trained on good and the other on bad strategies. Orange bars count the Hamming distance between allocations for each task (where number tasks = 9) in both simulations the average Hamming distance per task was 0.21, corresponding to an average difference of 2 positions in X. Blue bars measure the percent of task allocations issued by good and bad models that are the same (~50%). Purple bars indicate the percentage of predictions from good and bad models that are the same for all tasks (5%). Red indicates the percent of null predictions. Green indicates the average silhouette score per task, computed from predictions QX from good and bad models. Right, a second simulation, training two models on good and bad data and testing on a common dataset.

When model predictions were explored at the level of aggregated traits, it was found that the model would only allocate vehicles to the most salient tasks (Fig. 15). Like the aggregated traits, taking silhouette scores of model predictions reveal the model only strongly distinguishes between three or four tasks of the nine (Fig. 16). The closest thing to an explanation is that smaller signals (e.g., Fig. 9) tend to not translate into allocations. Figure 17 shows that agent utilization can vary between simulations.

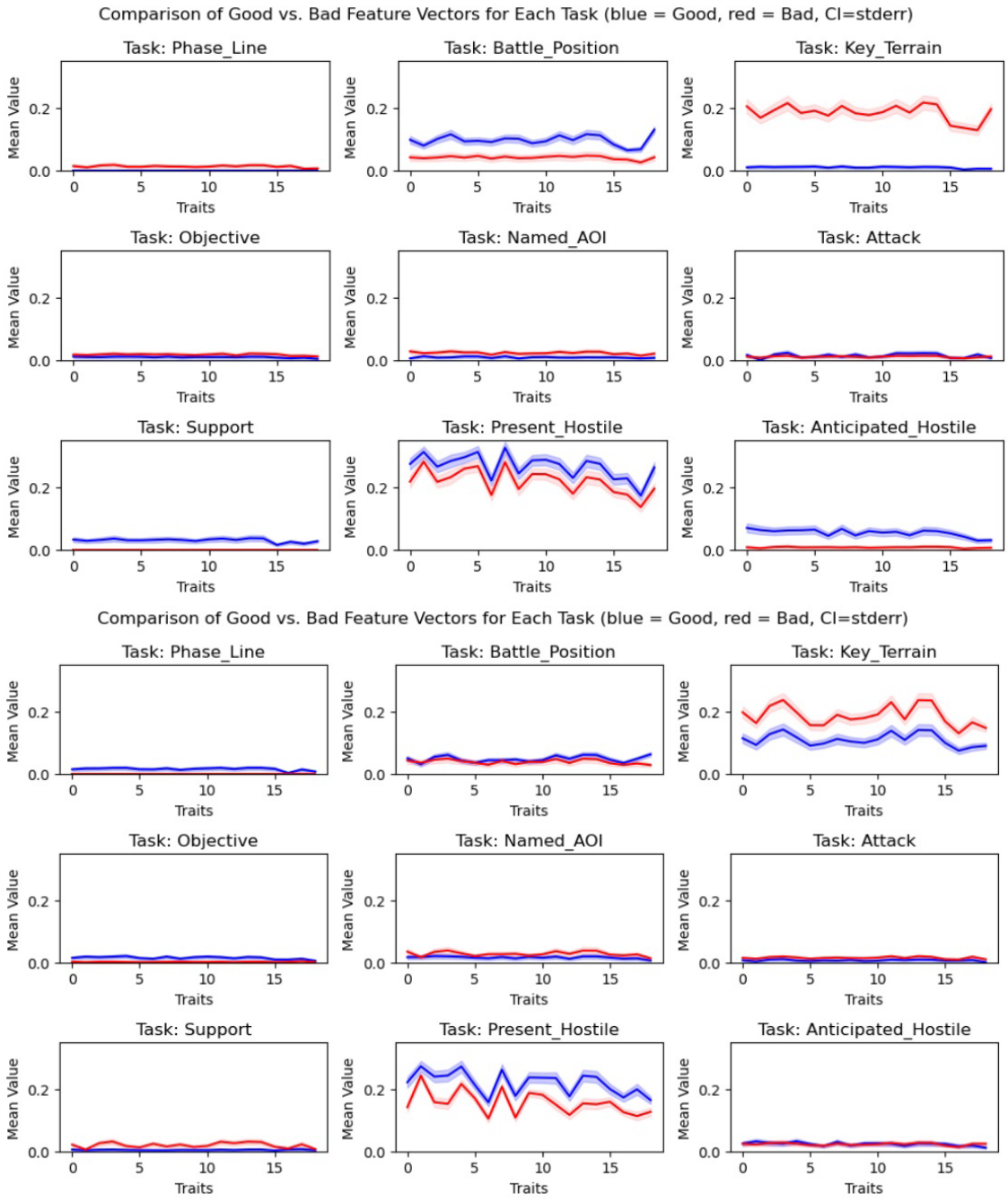

Fig. 15 Outputs for models trained on good (blue) and bad (red) strategies differ. Top and bottom, two simulations are shown that correspond to Figs. 8, 9, 13, and 14. Each figure compares the output XQ of a model trained on good strategies (blue) vs. a model trained on bad strategies (red) given the same input to the model.

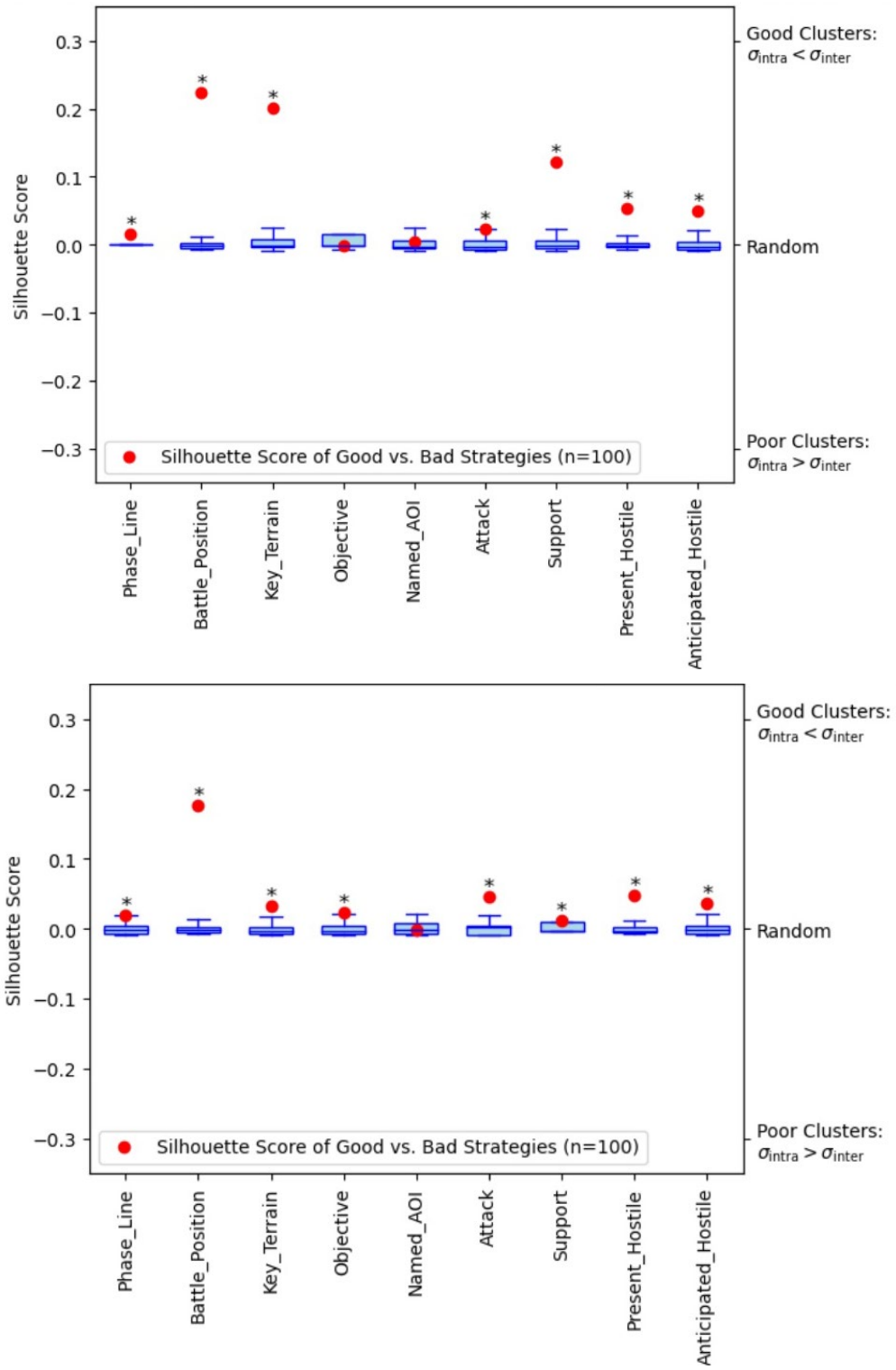

Fig. 16 Silhouette scores show outputs from models trained on good (blue) and bad (red) strategies differ (95% CI, 500 random labels). Top and bottom images correspond to different simulations and align with Figs. 8, 9 and 13–15. Red points chart the computed silhouette scores comparing the contrast (e.g., Fig. 12) in predictions from models tested on the same dataset but trained on different datasets. The blue box and whisker plots indicate the range and mean silhouette score from 500 random labelings of the same data.

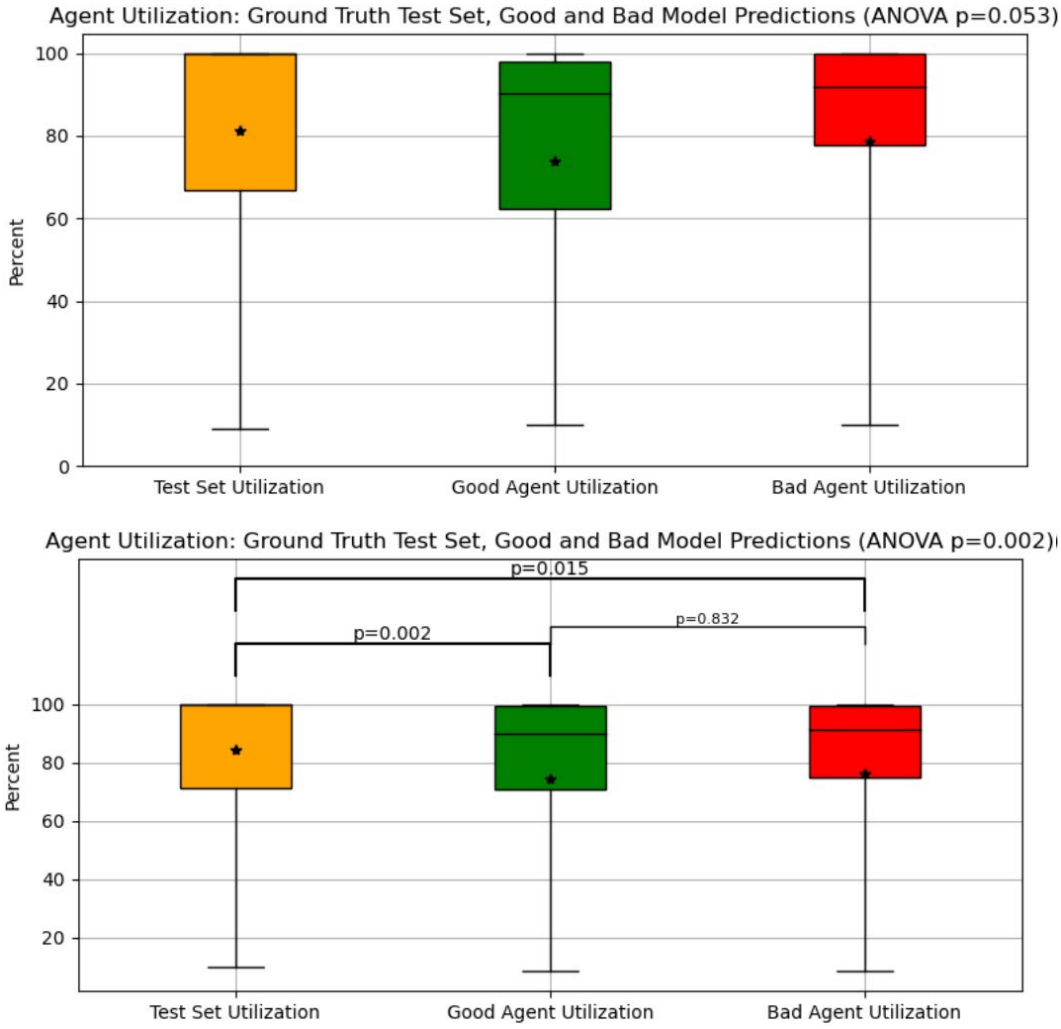

Fig. 17 Agent utilization test set vs. predictions by models trained on good and bad stochastic data. Two sets of plots representing different simulation experiments are shown (aligning with Figs. 8, 9, and 13–16). Orange bars indicate the agent utilization percentages in the ground truth set. Green bars indicate the agent utilization for models trained on good strategies. Red bars indicate agent utilization for models trained on bad strategies. Top, simulation with no significant differences between cases using ANOVA. Bottom, simulation with significant differences in agent utilization for all cases.

2.3.4.3 Discussion

2.3.4.3.1 Model solutions are dictated by shape of matrix Q. The model is built around a feature space for learning strategies using a fundamental equation: $Y = QX$ (X, Y, Q all matrices). Essentially, the model predicts strategies by solving for an unknown X when given Y, Q. Whether the system is underdetermined or overdetermined depends on the shape of Q whose dimensions are (S, U). The details of this argument are (1) $<N_s, Q>$ are given as inputs during testing (where N_s are constraints on the unknown X); (2) each iteration of the optimization algorithm gives the value of Y as one of the learned strategy centroids ${}_rC_m$ and solves for an

unknown X; and (3) $X \in \mathbf{Z}_{+}^{S\times M}$ and $Y \in \boldsymbol{R}_{+}^{M\times U}$ so the relative number of known and unknown variables is reflected by U and S, respectively, which are the dimensions of Q. Hence the ability of a search algorithm to find an optimal solution is influenced by the size of Q: when $S < U$, the system is overdetermined and there are only inexact solutions; when $S > U$ the system is underdetermined and an extra term may be needed for optimization to converge; when $S = U$ the system is determined and there is a unique inverse and a corresponding global minimum for the related convex optimization problem $||Y - XQ||^2$. If dimensionality is high ($M + U + S \rightarrow \infty$) solutions are more difficult to find. FY22 faced a situation where there were fewer traits than species ($S = 24, U = 11$, underdetermined) resulting in many possible solutions. Such issues can be solved in convex computing by introducing an additional regularization term in the optimization equation that would pick the X with the smallest or largest norm from the solution set satisfying $||Y - XQ||^2$. This was not done in the current study. FY24 faced the opposite problem: there were more traits than species ($S = 12, U = 19$) and the system was overdetermined. Such a system is almost always inconsistent and requires an approximate solution be found using the convex optimization problem $||Y - XQ||^2$. FY24 observed a large variability of outputs with the model failing to converge more than 25% of the time and failing to find exact solutions. Tests comparing different objective functions (Frobenius norm vs. sum of squared error [SSE]) and optimization toolboxes (docplex vs. cvxpy) were done using unconstrained minimization on some example problems. Overall, SSE with docplex performed best, but was not deployed in this study (it is available in the jupyter report).

2.3.4.3.2 GT-RSA does not describe how to adapt to a changing battle map. A limitation in the model was observed in FY24 relating to the fact the model does not have an explicit game map representation. The model must be adjusted prior to running on each battle map, otherwise the model runs the risk of "ghost tasking" or assigning tasks to vehicles that are impossible for that battlespace (e.g., attacking a nonexistent hostile). This flaw means that past learning may not transfer to new situations. One solution to this rests in the observation that the task subspaces for aggregated traits and their associated strategy clusters are learned independently and are more or less modular. This means the model can be adapted offline to a changing game map by selecting tasks that match the new battlespace and building the model accordingly. This introduces issues related to the management of pretrained task modules and the offline identification of which task modules are needed to complete a mission. Another possibility, which was applied in a limited fashion in the present study, is to incorporate context and dynamic variables into Q so that allocations are context-sensitive. This approach may require adding or removing terms related to having an underdetermined system.

2.3.4.3.3 Comparison of FY24 and FY22 results. The FY22 application of the GT-RSA algorithm to the low-level problem saw much more accuracy (exact matches to ground truth) than was seen in FY24 (95% vs. ~80% [as estimated using the complement of the FY24 Hamming distance]). It seems likely this difference originates in the data; however, FY22 INFORMS data was not available to confirm.

2.3.4.3.4 FY24 experiment 1 violated GT-RSA assumptions. The first experiment relied on expert-generated good and bad battle scenarios; however, this data was created in a story-like fashion and the consequences of different strategies unfolded and amplified across time. A training set where consequences are separated from the current game state probably violates the single-task multiple-robot informed allocation (STMRIA) assumptions of the model. STMRIA constraints require each robot can execute at most one task at a time; some tasks require multiple robots; and tasks are allocated to robots based on current information without planning for future allocations. Experiment 2 addressed this problem by identifying strategy clusters that were the most different in good and bad battle scenarios. These strategy clusters were used to create a new training set by seeding a multivariate Gaussian. When trained on this data, the model predicted different allocation matrices X_{good}, X_{bad}.

2.3.4.3.5 GT-RSA is not quite a reinforcement learning algorithm. The model does not provide an explicit mechanism for ranking scenarios by outcome value, nor does it have a reinforcement signal. The only similar mechanism comes from the offline selection of optimal strategies to use for training. It is difficult to see how this qualifies the algorithm as a reinforcement learning model while excluding classification algorithms with handpicked/supervised training sets. That said, Eq. 4 looks like it could serve as a task-based value function (utility). If this were made an explicit assumption of the model, then the model could be considered a type of reinforcement-learning model that learns value functions for tasks using unsupervised clustering.

2.3.4.3.6 Training and testing methods quench weaker trends. The model learns real-valued cluster centroids in the training phase. However, it uses these real-valued centroids during testing to output sparse (binary) positive integer matrices, X. Specifically, it finds a sparse binary assignment matrix X that minimizes the error $Y - QX$, where Y is positive real and Q is bound $(0, 100)$. This necessarily results in one-to-many compressions of information that makes the model insensitive to small differences in Y and Q, resulting in a model that captures strong trends and quenches weak ones.

2.4 Conclusions

The primary benefit of the model is in its analysis of the STMRIA problem. It provides a formalism that allows the problem to be discussed in a principled way that can lead to engineering solutions. While this model is not robust, the supporting theory makes it easy to extend and resolve performance issues.

The ability of the model to generate useful predictions is strongly linked to the human screening of training data and identification of traits. If a species-trait matrix Q describes more traits than tasks, then the unknown allocation matrix X might be highly variable. If the species-trait matrix Q has fewer traits (i.e., is square or underdetermined), it might not be context-sensitive and may require additional terms for optimization.

3. A-Priori, Optimization-Driven Allocation Model

The allocation of resources in the mounted platoon is often broken into high- and low-level problems. The high-level problem addresses the tasking of vehicles in the mission. The low-level problem addresses the assignment of crew and autonomy to vehicle subsystems (e.g., mobility, lethality, slewable sensor). The UT algorithm gives a coupled solution to the high- and low-level mounted platoon problems. The UT algorithm defines a value function for allocations at each level and uses a greedy search algorithm to find the allocation that maximizes the combined value. A proof rooted in matroid theory demonstrates the method converges to find good (but not optimal) solutions to the allocation problem.

3.1. Overview of UT Algorithm and Theory

The UT algorithm collects all possible high-level taskings into a ground set E and all possible low-level taskings into the ground set V. The algorithm generates a solution to the mounted platoon problem by selecting two allocation sets from these ground sets: an allocation set A for high-level taskings and an allocation set B for low-level taskings. A greedy search algorithm builds the allocation sets A and B element-wise by selecting elements from E and V that maximize the value of a coupled equation:

$$h(A,B) = g(A) + f(A,B)\,, \tag{10}$$

where $g(A)$ is a value (utility) function for the high-level and $f(A,B)$ is a value function for the low-level problem. The stability and convergence properties of the algorithm are shown as an application of submodular functions and matroid theory. As applied to the current problem, a uniform matroid is a set theory structure

(having set operators and cardinality) with four defining properties: 1) a ground set *E* that describes the universe of taskings, 2) a family of independent subsets *I* that describe possible allocations, 3) a ranking parameter *k* that gives the maximal size of an allocation set in *I* (sets larger than *k* are considered dependent), and 4) an augmentation property that describes how to build families of independent subsets in an element-wise manner. The notation for a uniform matroid uses a ground set *E*, rank parameter *k*, and *I* to represent the family of independent subsets:

$$M_{k,E} \equiv < E, I > . \tag{11}$$

3.1.1 High-Level Matroid

The ground set *E* is the collection of all possible high-level task assignments and consists of (vehicle, task) pairs which are drawn from the sets M and N:

M = {MCV1, MCV4, RCV2, RCV3, RCV5, RCV6}

N = {Battle Position, Detected, Objective, NAI, Suspected, Phase Line, Key Terrain, Move}.

Here $|N| = 8$ and $|M| = 6$ and vehicles and tasks are defined in Tables 3 and 4, respectively, with NAI referring to Named AOI (Table 4) and Move as a new task. With these definitions, the ground set E consisted of 48 elements that represent possible taskings.

The solution to the high-level problem is an allocation set A drawn from the ground set E. For the high-level problem, the allocation set A is an independent set of taskings drawn from E. The allocation set A is an independent set of E whose maximal size equal to the number of vehicles in E reflecting STMRIA constraints. With rank parameter $k = |M|$, the set of possible allocations sets can be shown to be a family of independent subsets. In this application, the constraints for *A* are

$$\{\mathrm{A} \subseteq \mathcal{P}(\mathrm{E}) \mid \forall(\mathrm{x}_1, \mathrm{y}_1), (\mathrm{x}_2, \mathrm{y}_2) \in \mathrm{A},\ \mathrm{x}_1 \neq \mathrm{x}_2 \quad \text{and} \quad rank(A) \leq m\}, \tag{12}$$

where *A* is an independent subset in the powerset of *E,* each vehicle can only be assigned one task, and the cardinality of an allocation set is $|A| \leq k = |M| = 6$.

3.1.2 Low-Level Matroid

The ground set V represents all possible low-level assignments for crew and autonomies. These (crew, subsystem) pairs are drawn from the sets P and Q:

P = {CS1, CS2, CS3, CS4, CS5, CS6, CS7, AUS1, AUS2, AUS3, AUD1, AUD2, AUD3, CS8, CS9, CS10, CS11, CS12, CS13, CS14, AUS4, AUS5, AUS6, AUD4, AUD5, AUD6}

Q = [MCV1_Mobility, RCV2_Mobility, RCV3_Mobility, MCV1_Primary, MCV1_Auxiliary, RCV2_Primary, RCV2_Auxiliary, RCV3_Primary, RCV3_Auxiliary, UAV1, UAV2, UAV3, MCV4_Mobility, RCV5_Mobility, RCV6_Mobility, MCV4_Primary, MCV4_Auxiliary, RCV5_Primary, RCV5_Auxiliary, RCV6_Primary, RCV6_Auxiliary, UAV4, UAV5, UAV6],

where crew and autonomy are defined in Table 6, vehicle subsystems are defined in Table 7, and $|P| = 26$, $|Q| = 24$.

Table 6. Crew and autonomy for low-level problem

Crew and autonomy	
Team 1: paired to MCV1, RCV2–3, UAV1–3	Team 2: assigned to MCV4, RCV5–6, UAV4–6
CS1	CS8
CS2	CS9
CS3	CS10
CS4	CS11
CS5	CS12
CS6	CS13
CS7	CS14
Autonomies paired to MCV1, RCV2–3	Autonomies paired to MCV4, RCV5–6
AUS1–3: Primary, auxiliary only	AUS4–6: Primary, auxiliary only
AUD1–3: Mobility only	AUD4–6: Mobility only

Table 7. Species and vehicle subsystems for low-level problem

Vehicle species and subsystems		
MCV1-Primary	9. RCV3-Primary	17. RCV5-Primary
MCV1-Auxiliary	10. RCV3-Mobility	18. RCV5-Mobility
MCV1-Mobility	11. RCV3-Auxiliary	19. RCV5-Auxiliary
MCV1-UAV	12. RCV3-UAV	20. RCV5-UAV
RCV2-Primary	13. MCV4-Primary	21. RCV6-Primary
RCV2-Mobility	14. MCV4-Mobility	22. RCV6-Mobility
RCV2-Auxiliary	15. MCV4-Auxiliary	23. RCV6-Auxiliary
RCV2-UAV	16. MCV4-UAV	24. RCV6-UAV

The possible assignments of crew to subsystems in the ground set V are not as simple to enumerate as in the high-level problem as there are added constraints. First, crew and autonomies are divided into two teams; second, while human crew can be assigned to any vehicle subsystem within a team, certain autonomies can only be assigned to mobility (driving autonomy; AUD) or primary and auxiliary subsystems (sensing autonomy; AUS). These constraints result in a low-level ground set V composed of 186 ordered pairs describing possible assignments of crew and autonomy to subsystems.

The allocation set B, drawn from the ground set V, represents the solution to the low-level problem. The size of B is constrained by bijective or single-task single-robot informed allocation (STSRIA) constraints. In this, application the constraints for B are

$$\{B \subseteq \mathcal{P}(V) \mid \forall (x_1, y_1), (x_2, y_2) \in B,\ x_1 \neq x_2,\ y_1 \neq y_2 \text{ and } rank(B) \leq min(p, q)\}, \quad (13)$$

where B is an independent subset in the powerset of V and the bijective constraints mean the number of elements in B is limited by the smaller of the sets P $(n = 26)$ and Q $(n = 24)$. Hence, $|B| \leq k = 24$.

3.1.3 High-Level Value Function

In addition to STMRIA constraints and rank, a value function provides a third constraint on how the allocation set A is selected from the ground set E. The high-level value function is derived from the static weapon target assignment (SWTA) problem. SWTA addresses the case where there are a number of assets and targets, and each asset is assigned a single target. The SWTA problem requires each target be assigned a value or priority and the probability that each asset can successfully complete its task be known. As adapted to the high-level problem, the value function g maps allocations A to real-valued scalars ($g: 2^E \rightarrow \boldsymbol{R}^+$):

$$g(A) = \sum_{n=1}^{|N|} v_n \left[1 - \prod_{\{m:(m,n)\in A\}} \left(1 - P_{n,m}\right)\right], \quad (14)$$

where A is the allocation set for the high-level problem and m, n denote the set of high-level vehicles and tasks, respectively. Each high-level task has a value $v_i \geq 0, \forall i \in [1, |N|]$ that corresponds to SWTA target value. $P_{n,m} \in [0,1]$ denote the probability that vehicle n completes high-level task m.

3.1.4 Low-Level Value Function

In addition to STSRIA and rank, a low-level value function impacts the selection of taskings from the ground set V to the allocation set B. The low-level value function $f(A, B)$ is a monotone function that maps allocation matrices to a real-valued scalar ($f: 2^E \times 2^V \rightarrow \boldsymbol{R}^+$). The function is a linear equation involving two sets of matrices that are multiplied element-wise:

$$f(A, B) = \sum_{\{(p,q):(p,q)\in B\}} \left[w(A) \odot C_{p,q}\right], \quad (15)$$

where the matrix $C \in \boldsymbol{R}_+^{|Q|\times|P|}$ represents the utilities (efficiencies) with which various crew perform various assignments and $w(A)$ is a monotone function that

maps high-level taskings to a matrix $w: 2^E \rightarrow \boldsymbol{R}_+^{|Q|\times|P|}$. After element-wise multiplication, the values of represented members of B are summed. This is implemented as a summation of the element-wise multiplication of $w(A)$ and C, masked by binary matrix B.

3.1.5 Coupled Greedy Search Algorithm

The coupled greedy search algorithm (Fig. 18) has three nested for-loops with an approximate time complexity of $O(n^2)$. The outer loop cycles through $|E|$ loops (however, in practice the outer loop terminates after at most $k = |M|$ loops are run). Each iteration of the outer loop examines the solutions returned by the inner loops and keeps the best. The middle loop cycles through the families of independent subsets in $\mathcal{P}(E)$ until the constraints are violated (STMRIA, rank). Each valid independent subset A' is sent from the middle loop to the inner loop. The inner loop takes these candidate allocation matrices A' and cycles through the family of independent subsets of $\mathcal{P}(V)$, creating candidate allocations B' and computing the low-level value function *f(A', B')*. For each A', the inner loop returns the highest scoring candidate B'. The middle loop returns a set of paired candidate allocation matrices (A', B') to the outer loop, which are then scored using Eq. 10 with the highest scoring allocations sets returned.

```
def greedy_wta_lta(E: set, V: set, g, f, w) -> tuple:
A = set()
A_G = set()
B_G = set()

for _ in range(len(E)):
    B_set = set()

    for a in E - A:
        if is_independent(A.union({a}), E ):
            B = set()
            V_prime = set()

            for _ in range(len(V)):
                b_prime = max(V - V_prime, key=lambda b: f(A.union({a}),
                ↪ B.union({b})))

                if is_independent(B.union({b_prime}), V):
                    B.add(b_prime)

                V_prime.add(b_prime)

            B_set.add((a, frozenset(B)))

    if not B_set:
        break

    a_prime, B_G = max(B_set, key=lambda a_B: d(A, a_B[0], a_B[1], g, f))
    A.add(a_prime)

A_G = A
return A_G, B_G
```

Fig. 18 Pythonic pseudocode for UT greedy search algorithm. In application, two separate is_independent() functions are deployed: is_independent_STMRIA and is_independent_STSRIA.

3.2 Simulations

Run-time properties of the algorithm were assessed in a simulation using expert demo data and where high-level tasks were equally weighted. The algorithm was interrupted after 10, 24, and 48 iterations to explore execution time and the consistency of taskings.

3.3 Methods

Algorithm was implemented using Python and csv data files. Code and data are maintained in a sitcore gitlab repository.[†] Model parameters are set out in config.py, with value function parameters in separate csv files, high-level probabilities are in Pmn_upper2.csv, matrix Cpq is computed from low-level utilities in Cpq_lower.csv, and w(A) weightings are computed from Cpq_lower.csv. Results are displayed in a jupyter report UT_test.ipynb that links to images in a ./img subdirectory. Simulations were performed using a laptop with a 2.5 GHz i7 CPU and 32 GB of RAM.

3.3.1 Instantiation of UT Algorithm for Mounted Platoon

The ground set E references the vehicles and tasks in Tables 3 and 4. Fully enumerated, the ground set E consisted of 48 taskings:

E = {(MCV1, Battle Position), (MCV1, Detected), (MCV1, Objective), (MCV1, NAI), (MCV1, Suspected), (MCV1, Phase Line), (MCV1, Key Terrain), (MCV1, Move), (MCV4, Battle Position), (MCV4, Detected), (MCV4, Objective), (MCV4, NAI), (MCV4, Suspected), (MCV4, Phase Line), (MCV4, Key Terrain), (MCV4, Move), (RCV2, Battle Position), (RCV2, Detected), (RCV2, Objective), (RCV2, NAI), (RCV2, Suspected), (RCV2, Phase Line), (RCV2, Key Terrain), (RCV2, Move), (RCV3, Battle Position), (RCV3, Detected), (RCV3, Objective), (RCV3, NAI), (RCV3, Suspected), (RCV3, Phase Line), (RCV3, Key Terrain), (RCV3, Move), (RCV5, Battle Position), (RCV5, Detected), (RCV5, Objective), (RCV5, NAI), (RCV5, Suspected), (RCV5, Phase Line), (RCV5, Key Terrain), (RCV5, Move), (RCV6, Battle Position), (RCV6, Detected), (RCV6, Objective), (RCV6, NAI), (RCV6, Suspected), (RCV6, Phase Line), (RCV6, Key Terrain), (RCV6, Move)}

The ground set V referenced the crew and autonomies in Table 6 and vehicle subsystems in Table 7. Fully enumerated, the ground set V consists of 186 crew assignments:

V = [('AUS5', 'RCV5_Auxiliary'), ('AUS2', 'RCV2_Primary'), ('CS9', 'MCV4_Mobility'), ('CS14', 'RCV6_Auxiliary'), ('CS7', 'MCV1_Mobility'), ('AUS1', 'MCV1_Auxiliary'), ('CS10', 'MCV4_Auxiliary'), ('CS11', 'UAV6'), ('CS2', 'RCV2_Mobility'), ('CS4', 'RCV2_Primary'), ('AUS3', 'RCV3_Auxiliary'), ('CS13', 'RCV6_Primary'), ('AUD1', 'MCV1_Mobility'), ('AUD2', 'RCV2_Mobility'), ('CS2', 'RCV3_Primary'), ('CS7', 'RCV2_Mobility'), ('CS14', 'RCV5_Mobility'), ('CS10', 'RCV5_Auxiliary'), ('CS10', 'RCV6_Primary'), ('AUS6', 'RCV6_Primary'), ('CS4', 'RCV3_Primary'), ('CS14', 'RCV5_Primary'), ('AUS4', 'MCV4_Auxiliary'), ('AUS2', 'RCV2_Auxiliary'), ('CS1', 'MCV1_Auxiliary'), ('CS6', 'RCV2_Primary'), ('CS9', 'RCV5_Primary'), ('CS5', 'UAV2'), ('AUS3', 'RCV3_Primary'), ('CS6', 'RCV2_Auxiliary'), ('CS4', 'UAV2'), ('CS13', 'RCV6_Auxiliary'), ('CS8', 'RCV5_Auxiliary'), ('CS8', 'MCV4_Primary'), ('CS8', 'RCV5_Primary'), ('AUS5', 'RCV5_Primary'), ('AUD4', 'MCV4_Mobility'), ('CS9', 'MCV4_Primary'), ('AUD6', 'RCV6_Mobility'), ('CS6', 'MCV1_Auxiliary'), ('CS9', 'UAV5'),

[†] https://gitlab.sitcore.net/arl/hred/hat-erp/project-6/texas-arl-ca.git

('CS12', 'RCV5_Auxiliary'), ('CS5', 'RCV2_Auxiliary'), ('CS6', 'RCV3_Mobility'), ('CS13', 'RCV6_Mobility'), ('CS3', 'RCV2_Primary'), ('CS4', 'MCV1_Primary'), ('CS6', 'RCV2_Mobility'), ('CS5', 'RCV2_Primary'), ('AUD3', 'RCV3_Mobility'), ('CS1', 'RCV2_Mobility'), ('AUS4', 'MCV4_Primary'), ('CS5', 'RCV3_Auxiliary'), ('CS4', 'RCV2_Auxiliary'), ('CS3', 'UAV1'), ('CS6', 'UAV2'), ('CS3', 'UAV2'), ('CS11', 'RCV6_Primary'), ('CS4', 'RCV2_Mobility'), ('CS10', 'RCV6_Auxiliary'), ('CS14', 'MCV4_Primary'), ('CS9', 'RCV6_Mobility'), ('CS11', 'RCV5_Primary'), ('CS1', 'RCV3_Primary'), ('CS4', 'RCV3_Auxiliary'), ('CS6', 'MCV1_Mobility'), ('CS9', 'RCV5_Mobility'), ('CS8', 'RCV6_Auxiliary'), ('CS2', 'MCV1_Auxiliary'), ('CS10', 'RCV5_Mobility'), ('CS10', 'RCV5_Primary'), ('CS8', 'RCV6_Mobility'), ('CS13', 'MCV4_Mobility'), ('CS2', 'RCV2_Auxiliary'), ('CS11', 'RCV5_Mobility'), ('CS3', 'MCV1_Mobility'), ('AUS6', 'RCV6_Auxiliary'), ('AUS1', 'MCV1_Primary'), ('CS7', 'RCV3_Mobility'), ('CS1', 'RCV3_Mobility'), ('CS3', 'RCV3_Mobility'), ('CS11', 'MCV4_Mobility'), ('CS6', 'UAV1'), ('CS5', 'MCV1_Primary'), ('CS7', 'RCV2_Auxiliary'), ('CS12', 'UAV6'), ('CS12', 'MCV4_Auxiliary'), ('CS1', 'UAV3'), ('CS8', 'UAV4'), ('CS14', 'UAV5'), ('CS9', 'RCV6_Primary'), ('CS1', 'MCV1_Mobility'), ('CS3', 'RCV3_Primary'), ('CS12', 'UAV5'), ('AUD5', 'RCV5_Mobility'), ('CS5', 'MCV1_Auxiliary'), ('CS14', 'RCV6_Mobility'), ('CS3', 'UAV3'), ('CS12', 'MCV4_Primary'), ('CS8', 'RCV6_Primary'), ('CS4', 'RCV3_Mobility'), ('CS8', 'RCV5_Mobility'), ('CS10', 'UAV5'), ('CS3', 'MCV1_Primary'), ('CS13', 'RCV5_Primary'), ('CS2', 'RCV3_Auxiliary'), ('CS9', 'MCV4_Auxiliary'), ('CS5', 'UAV1'), ('CS1', 'RCV3_Auxiliary'), ('CS4', 'UAV3'), ('CS10', 'RCV6_Mobility'), ('CS8', 'MCV4_Auxiliary'), ('CS13', 'MCV4_Primary'), ('CS8', 'MCV4_Mobility'), ('CS8', 'UAV6'), ('CS12', 'RCV6_Mobility'), ('CS5', 'UAV3'), ('CS12', 'RCV5_Primary'), ('CS2', 'RCV2_Primary'), ('CS9', 'RCV6_Auxiliary'), ('CS1', 'RCV2_Auxiliary'), ('CS14', 'RCV5_Auxiliary'), ('CS14', 'UAV4'), ('CS7', 'UAV3'), ('CS11', 'RCV6_Auxiliary'), ('CS5', 'MCV1_Mobility'), ('CS14', 'UAV6'), ('CS10', 'MCV4_Mobility' ('CS2', 'MCV1_Primary'), ('CS8', 'UAV5'), ('CS3', 'RCV2_Auxiliary'), ('CS3', 'MCV1_Auxiliary'), ('CS11', 'UAV4'), ('CS6', 'MCV1_Primary'), ('CS2', 'UAV2'), ('CS6', 'RCV3_Primary'), ('CS7', 'UAV1'), ('CS12', 'UAV4'), ('AUS3', 'RCV2_Primary'), ('CS4', 'UAV1'), ('CS2', 'MCV1_Mobility'), ('CS12', 'RCV6_Primary'), ('CS14', 'MCV4_Auxiliary'), ('CS14', 'MCV4_Mobility'), ('CS9', 'UAV4'), ('CS10', 'UAV4'), ('CS2', 'UAV1'), ('CS3', 'RCV2_Mobility'), ('CS7', 'MCV1_Auxiliary'), ('CS4', 'MCV1_Mobility'), ('CS7', 'UAV2'), ('CS13', 'UAV5'), ('CS1', 'MCV1_Primary'), ('CS14', 'RCV6_Primary'), ('CS4', 'MCV1_Auxiliary'), ('CS2', 'UAV3'), ('CS6', 'UAV3'), ('CS5', 'RCV3_Primary'), ('CS2', 'RCV3_Mobility'), ('CS3', 'RCV3_Auxiliary'), ('CS13', 'RCV5_Mobility'), ('CS13', 'RCV5_Auxiliary'), ('CS7', 'RCV3_Auxiliary'), ('CS1', 'UAV1'), ('CS10', 'MCV4_Primary'), ('CS13', 'MCV4_Auxiliary'), ('CS7', 'MCV1_Primary'), ('CS11', 'RCV5_Auxiliary'), ('CS1', 'RCV2_Primary'), ('CS7', 'RCV2_Primary'), ('CS12', 'RCV6_Auxiliary'), ('CS12', 'RCV5_Mobility'), ('CS5', 'RCV3_Mobility'), ('CS11', 'MCV4_Primary'), ('CS9', 'UAV6'), ('CS1', 'UAV2'), ('CS11', 'MCV4_Auxiliary'), ('CS9', 'RCV5_Auxiliary'), ('CS11', 'UAV5'), ('CS5', 'RCV2_Mobility'), ('CS13', 'UAV6'), ('CS13', 'UAV4'), ('CS10', 'UAV6'), ('CS12', 'MCV4_Mobility'), ('CS11', 'RCV6_Mobility'), ('CS7', 'RCV3_Primary'), ('CS6', 'RCV3_Auxiliary')]

3.3.2 Value Function Parameters

High-level vehicles were assigned normalized efficacies compatible with the SWTA model (i.e., meet probability axioms). These scores were saved in the file Pmn_upper2.csv and are shown in Table 8.

Table 8 Assigned probabilities for high-level value function

Vehicles	Battle position	Detected	Objective	NAI	Phase line	Suspected	Key terrain	Move
MCV1	0.00	0.00	0.00	0.50	0.00	0.00	0.75	1.00
RCV2	0.25	0.50	1.00	0.25	0.00	0.00	0.25	0.90
RCV3	0.75	1.00	0.50	0.00	0.00	1.00	0.50	0.90
MCV4	0.25	1.00	0.25	0.75	0.00	0.50	0.00	1.00
RCV5	0.50	0.25	0.75	0.25	1.00	1.00	1.00	0.90
RCV6	1.00	1.00	0.75	0.50	1.00	0.25	0.75	0.90

Low-level value function parameters were derived from parameters in the file probabilities.xlsx (saved in Gitlab folder ./UT_report). This file detailed parameters

that were normalized over (0, 1) for a 3D array with indices: high-level tasks (N), vehicle subsystems (Q), and crew (P). The 3D array was linearized and saved as a multi-index table in the file Cpq_lower.csv. The low-level matrix Cpq was derived from these values by averaging over high-level tasks (M) (get_Cqp_from_multi()), ensuring values in Cpq are bound between $[0, 1]$. The low-level matrix corresponding to $w(A)$ is computed on the fly in the inner loop of the greedy search algorithm using these same values and the current high-level candidate A'. The function get $wA()$ performs a weighted sum based on the number of vehicles assigned to each high-level task and is not bound in the $[0, 1]$ interval.

3.3.3 Modifications to UT Greedy Algorithm

Minor modifications were made to the UT algorithm. The function is_independent() varies between high- and low-levels so it was replaced by two functions (is_independent_STMRIA and is_independent_STSRIA). These different functions implement different standards for determining the parameter k (STMRIA/STSRIA) that fixes the maximum size of independent sets.

The second modification was to interrupt the algorithm after a fixed number of iterations. This had the effect of reducing the number of elements in the high-level ground set that were searched for maximal value. The algorithm was interrupted after 10, 24, or 48 iterations, the last of which represented the complete high-level ground set.

3.4 Results and Discussion

The greedy search algorithm was run with uniform weighting of high-level tasks. The algorithm was interrupted after 10 outer loop iterations, 24 iterations, and then 48 (all iterations). The final allocation sets A and B are detailed below and all three are show diagrammatically in Fig. 19.

A:{('RCV6', 'Objective'), ('RCV3', 'KeyTerrain'), ('RCV2', 'Move'), ('MCV1', 'Suspected'), ('RCV5', 'PhaseLine'), ('MCV4', 'NAI')}

B:{('CS1', 'UAV1'), ('CS2','RCV2_Mobility'), ('CS3', 'MCV1_Mobility'), ('CS4', 'MCV1_Primary'), ('CS5', 'RCV2_Primary'), ('CS6', 'RCV2_Auxiliary'), ('CS7', 'RCV3_Auxiliary'), ('AUS2', 'MCV1_Auxiliary'), ('AUS3', 'RCV3_Primary'), ('AUD1', 'RCV3_Mobility '), ('CS8', 'UAV4'), ('CS9', 'RCV5_Mobility '), ('CS10', 'MCV4_Mobility'), ('CS11', 'MCV4_Primary'), ('CS12', 'UAV6'), ('CS13', 'UAV5'), ('CS14', 'RCV5_Primary'), ('AUS4', 'RCV6_Primary'), ('AUS5', 'RCV5_Auxiliary'), ('AUS6', 'MCV4_Auxiliary'), ('AUD4', 'RCV6_Mobility')}

The run time of the algorithm was also explored. The algorithm appeared to scale linearly with the amount of the ground set E that was being explored (Fig. 20). This is possible because in addition to limitations on the outer loop that only iterates up to k_{upper}, the middle loop iterates over fewer taskings after every outer loop

because the number of available taskings decreases. Fig. 20 also shows that the total value (Eq. 10) increases as more of the ground set E is searched.

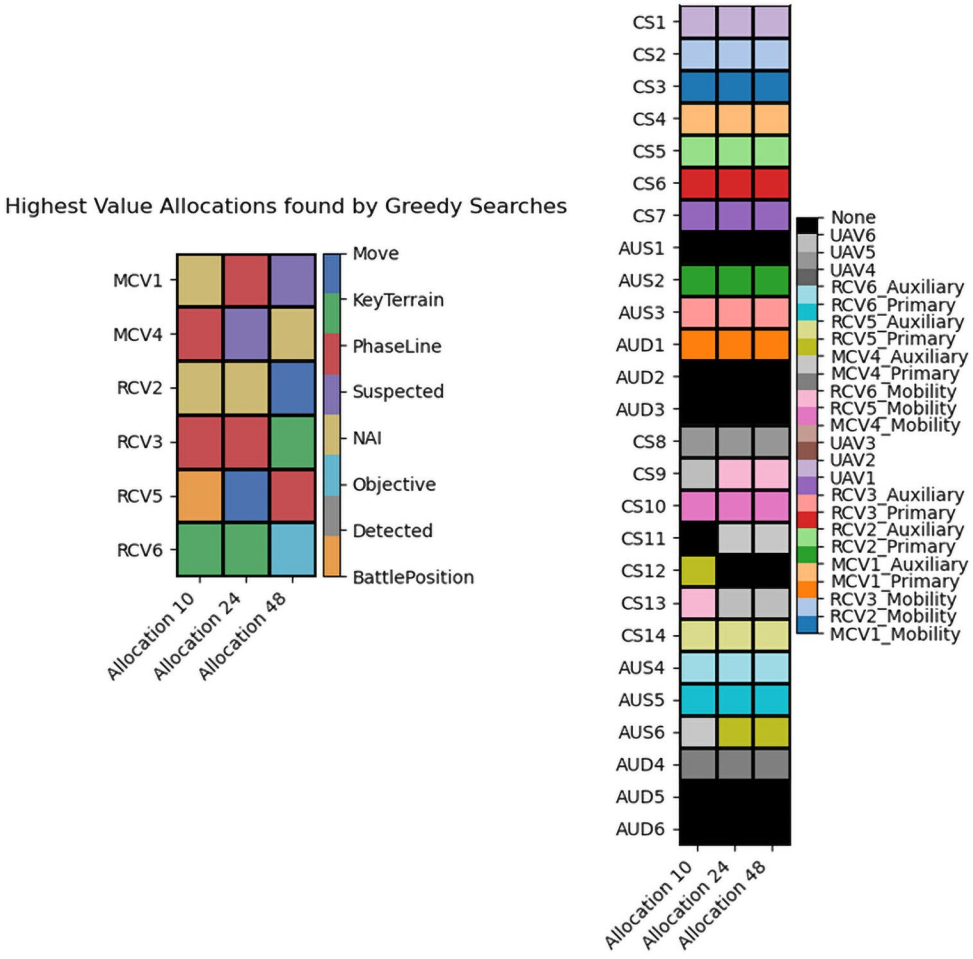

Fig. 19 High- and low-level allocations change with amount of ground set explored. Combinatorial outcomes for allocation sets A and B at different interruption times in the greedy search algorithm (10, 24, and 48) are shown on the left and right, respectively. Columns represent the allocations; x-axis describes the interruption point at which the allocation was determined. (Left) High-level vehicle-task allocations |A| = 6 (legend provides key for mapping colors to tasks, |N| = 8, y-axis indicates vehicles, |M| = 6). (Right) Low-level crew-subsystem assignments (|B |= 21) with legend mapping colors to subsystems (|Q| = 24) and crew and autonomy as indicated along y-axis (|P| = 26).

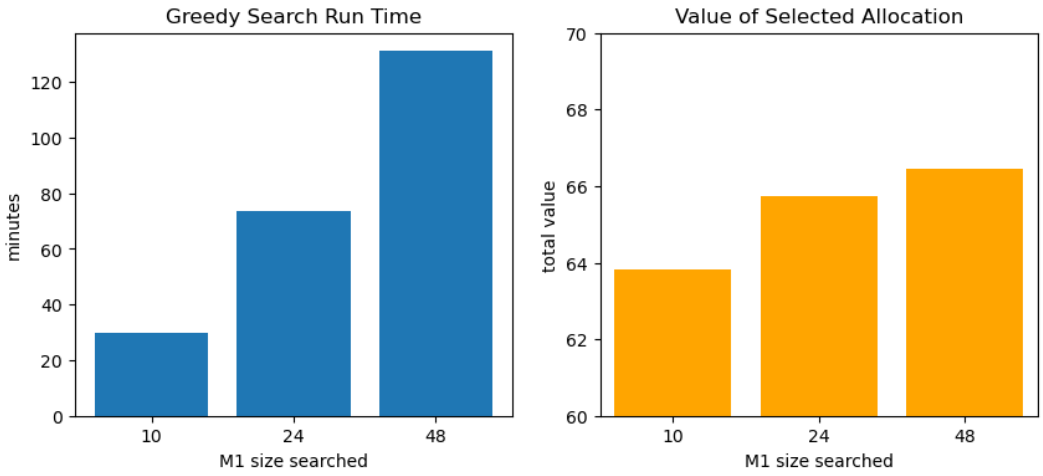

Fig. 20 Run time, total value of allocations vs. high-level ground set size. (Left) Run time for the greedy search algorithm was measured using a 2.5-GHz i7 CPU with 32 GB of RAM. X-axis indicates the number of taskings in the ground set that were searched before iterations were halted, |E| = 48. (Right) Total value (h[A, B], Eq. 10) associated with final allocations (A, B) output by algorithm.

Figure 21 charts out the number of times that the total value function is called. The number of function calls depends on the number of iterations through the middle and inner loops. While the inner loop must loop through all taskings in the ground set V on every call, the middle loop iterates through declining number of high-level taskings as STMRIA constraints eliminate redundant taskings.

3.5 Conclusions

The UT algorithm has the advantage that it is purpose-driven to solve the high- and low-mounted platoon problems simultaneously. Its methodology does not require black-box optimization toolboxes. The high-level value function is rather straightforward, and it would be easy to extract needed parameters from game statistics. A weakness in the model is in the low-level value function; specifically, the fact that $w(A)$ is very open-ended. It will take additional research to determine how to encode different battle scenarios into the function $w()$.

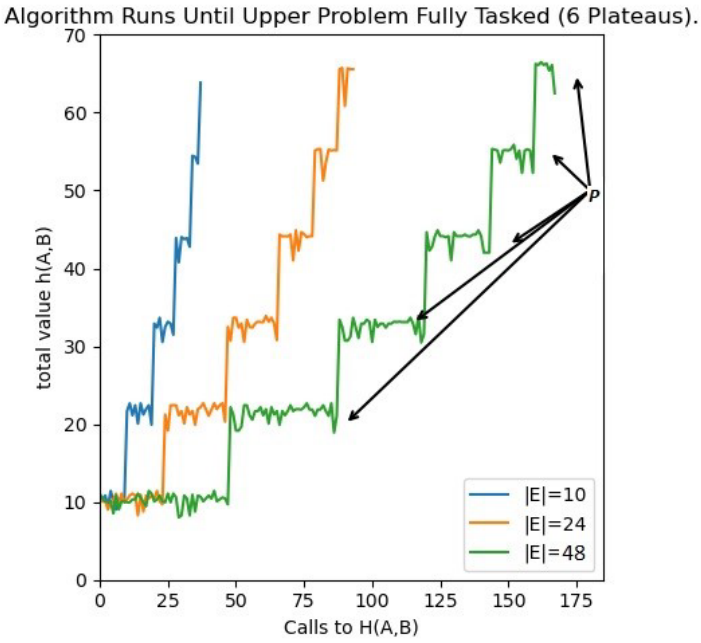

Fig. 21 Total value of candidate allocations computed at run time. Total value is plotted as the greedy search algorithm progresses. Total value is computed for candidates saved from the middle and inner loops. The number of plateaus (P) corresponds to the number of vehicles allocated to tasks in the high-level task. (Blue) Greedy search interrupted after 10 taskings in the ground set E have been searched. (Orange) Greedy search interrupted after 24 taskings in the ground set E have been searched. (Green) Greedy search interrupted after 48 taskings in the ground set E have been searched.

4. Discussion

4.1 Model Advantages and Disadvantages

It is easier to understand how to represent different game scenarios using simulation parameters with the GT-RSA model. The model is also relatively fast; it can be trained and tested on dozens of data samples in an hour or less. However, it must be run separately to generate solutions on both the high- and low-mounted platoon problems. GT-RSA computations do not feel fit for purpose; they likely will not scale well with the complexity of the future battlespace, they require a lot of offline work to condition training and testing data, and they require outside optimization toolboxes.

The UT algorithm is more logical than GT and offers coupled solutions to both high and low problems. However, based on the Python implementation of their general algorithm, generating one solution is significantly slower, taking 2 h. This relatively slow execution time can be attributed to Python's well-known inefficiency in handling nested “for” loops, which are prevalent in both the main algorithm and the marginal cardinality constraints of the UT approach. Therefore, the UT algorithm

could achieve solutions much faster if implemented in a low-level language, such as C/C++ or Fortran. However, the UT collaborators opted to use Python to facilitate a more straightforward translation from mathematical concepts to code. The main algorithm has a worst-case complexity of $\mathcal{O}(|E|^3 \cdot |V|^2)$, making it relatively efficient compared with other algorithms with guarantees of suboptimality. Another strength of the UT algorithm lies in its generalizability, as it can accommodate any submodular objective functions at both levels. Nevertheless, there is potential for further specialization. For instance, the lower-level STSRIA could be solved more efficiently using an off-the-shelf linear programming solver that employs specialized algorithms like the simplex method or interior point techniques.

4.2 Considerations When Choosing a Model

The UT algorithm is proven fit for purpose but slow when implemented in high-level languages such as Python. The GT-RSA algorithm requires time-consuming evaluation of parameters and conditioning of data, but it would probably be easier to downscale and yield something that would work in real time on standard personal computers.

5. References

Boutilier C, Brafman RI. Partial-order planning with concurrent interacting actions. Journal of Artificial Intelligence Research. 2001;14:105–146.

Capitan J, Spaan MT, Merino L, Ollero A. Decentralized multi-robot cooperation with auctioned POMDPs. International Journal of Robotics Research. 2013;32(6):650–671.

Choi H, Brunet L, How JP. Consensus-based decentralized auctions for robust task allocation. IEEE Transactions on Robotics. 2009;25(4):912–926.

Gerkey BP, Matarić MJ. A formal analysis and taxonomy of task allocation in multi-robot systems. International Journal of Robotics Research. 2004;23(9):939–954.

Gerkey BP, Matarić MJ. Sold!: auction methods for multirobot coordination. IEEE Transactions on Robotics and Automation. 2002;18(5):758–768.

Headquarters, Department of the Army. Tank platoon. Department of the Army (US); 2019 July. Army Techniques Publication No.: ATP 3-20.15.

Hirsch J, Neumayer M, Ponsar H, Kosak O, Reif W. Distributed constraint optimization for task allocation in self-adaptive manufacturing systems. Proceedings of the 2021 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C); 2021 Sep 27–Oct 1; virtual. c2021. p. 62–67.

Jones EG, Dias MB, Stentz A. Time-extended multirobot coordination for domains with intra-path constraints. Autonomous Robots. 2011;30:41–56.

Lee Y, Khalil A, Bakolas E, Gremillion G. A two-phase algorithm for joint task assignment and coordination in hierarchical multi-vehicle systems. Proceedings of the AIAA SciTech 2025 Forum; 2025 Jan 6–10; Orlando, FL. Forthcoming.

Macarthur KS, Stranders R, Ramchurn SD, Jennings NR. A distributed anytime algorithm for dynamic task allocation in multi-agent systems. Proceedings of the AAAI Conference on Artificial Intelligence. 2011;25(1):701–706.

Mohammad U, Sorour S, Hefeida M. Energy aware task allocation for semi-asynchronous mobile edge learning. IEEE Transactions on Green Communications and Networking. 2023;4(7):1766–1777.

Nanjanath M, Gini M. Repeated auctions for robust task execution by a robot team. Robotics and Autonomous Systems. 2010;58(7):900–909.

Nissim R, Brafman R. Distributed heuristic forward search for multi-agent planning. Journal of Artificial Intelligence Research. 2014;51:293–332.

Sarne D, Krau S. Solving the auction-based task allocation problem in an open environment. Proceedings of the AAAI Conference on Artificial Intelligence. 2005;20:164–169.

Shorinwa O, Yu J, Halsted T, Koufos A, Schwager M. Distributed multi-target tracking for autonomous vehicle fleets. IEEE International Conference. 2020;3495–3501.

Srikanthan A, Ravichandar H. Resource-aware adaptation of heterogeneous strategies for coalition formation; 2022 [updated 2022 Jan 25]. https://arxiv.org/abs/2108.02733.

Stanton N, Roberts A. Block off: an examination of new control room configurations and reduced crew sizes examining engineered production blocking. Cognition, Technology and Work. 2020;22:29–55.

Torreno A, Onaindia E, Sapena O. FMAP: distributed cooperative multi-agent planning. Applied Intelligence. 2014;41(2):606–626.

Tozicka J, Jakubuv J, Komenda A. PSM-based planners description for CoDMAP 2015 competition. Proceedings of the Competition of Distributed and Multi-Agent Planners (CoDMAP). 2015;29–32.

Wolpert DH, Wheeler KR, Tumer K. General principles of learning-based multi-agent systems. AGENTS ’99: Proceedings of the Third Annual Conference on Autonomous Agents. 1999;77–83.

Ye L, Yang Y, Meng W, Wu X, Li X, Zhu R. Offline and online task allocation algorithms for multiple UAVs in wireless sensor networks. EURASIP Journal on Advances in Signal Processing. 2024:21. https://doi.org/10.1186/s13634-024-01116-4.

Zlot R, Stentz A. Market-based multirobot coordination for complex tasks. The International Journal of Robotics Research 2006;25(1):73–101.

List of Symbols, Abbreviations, and Acronyms

3D	three-dimensional
ANOVA	analysis of variance
AOI	area of interest
AP	armor-piercing
AUD	driving autonomy
AUS	sensing autonomy
CFSTP	coalition formation with spatial and temporal constraints problem
CI	confidence interval
COAX	coaxial machine gun ammunition
COP	common operating picture
CPU	central processing unit
FMAP	forward multi-agent planning
FY22	fiscal year 2022
FY24	fiscal year 2024
GT	Georgia Institute of Technology
HE	high-explosive
INFORMS	Information for Mixed Squads
MAP	multi-agent planning
MCV	manned combat vehicle
MG	machine gun
NAI	named AOI
NGCV	next-generation combat vehicle
RAM	random access memory
RCV	robotic combat vehicle
RSA	resource aware allocation
SOP	standard operating procedure
SSE	sum of squared error

STMRIA	single-task multiple-robot informed allocation
STSRIA	single-task single-robot informed allocation
SWTA	static weapon target assignment
UAV	unmanned aerial vehicle
UT	the University of Texas at Austin
WLU	wonderful life utility
WMI	Warfighter machine interface

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
1 (PDF)	DEVCOM ARL FCDD RLB CI TECH LIB