Optimization-based Agent Simulations for Evaluating the SPEYES System[1]

Student Submission

(Modeling and Simulation)

Candra Meirina[2,*]
E-mail: meirina@engr.uconn.edu
Feili Yu[2,*]
E-mail: yu02001@engr.uconn.edu
Sui Ruan[2,*]
E-mail: sruan@engr.uconn.edu

Georgiy M. Levchuk
Aptima Inc.
12 Gill Street, Suite 1400
Woburn, MA 01801
Phone: (781) 935-3966 ext. 267
E-mail: georgiy@aptima.com

Krishna R. Pattipati[2,3]
[2]University of Connecticut, Dept. of Electrical and Computer Engineering
371 Fairfield Road, Unit 1157
Storrs, CT 06269-1157
Fax: 860-486-5585
Phone: 860-486-2890
E-mail: krishna@engr.uconn.edu

David L. Kleinman
C4I Academic Group
Naval Post-graduate School
589 Dyer Road, Monterey, CA 93943
Fax: 831-656-3679
Phone: 831-656-4148
E-mail: dlkleinm@nps.navy.mil

Robert L. Popp
Information Exploitation Office, DARPA
3701 N. Fairfax Drive, Arlington, VA 22203-1714
Phone: (703) 248-1520
E-mail: rpopp@darpa.mil

[1]This work was supported by the Information Exploitation Office of the Defense Advanced Research Projects Agency. This document carries the Distribution Statement "A" (Approved for Public Release, Distribution Unlimited).
[3]Correspondence: krishna@engr.uconn.edu
*Student

Optimization-based Agent Simulations for Evaluating the SPEYES System*

Candra Meirina[1], Feili Yu[1], Sui Ruan[1], Georgiy M. Levchuk[2], Krishna R. Pattipati[1,5], David L. Kleinman[3], Robert L. Popp[4]

[1]Univ. of Connecticut, Dept. of Electrical and Computer Engineering, Storrs, CT
[2]Aptima Inc., Woburn, MA
[3]Naval Postgraduate School, Dept. of Information Sciences , Monterey, CA
[4]Information Exploitation Office, DARPA, Arlington, VA

Abstract

This paper presents an optimization-based agent-driven Distributed Dynamic Decision-making (DDD) simulation model to evaluate the SPEYES (Security and Patrolling Enablers Yielding Effective SASO - Support and Stability Operations) system. The key challenge is to quantify the force multiplying effect of SPEYES technologies, which span sensing, situation awareness/command and control (SA/C^2), and shaping components. The performance improvements were measured in terms of timeliness, effectiveness, and efficiency of operations. The behaviors of optimization-based agents were first calibrated to those of human-in-the-loop simulations. The agent-driven simulation results indicated that integration of SPEYES sensing, SA/C^2, and shaping technologies provided significant performance improvements to the force across all measures. Even at 50%-reduced force, the SPEYES system maintained significant performance improvements over regular operations with a full force and without SPEYES, thus confirming the force multiplier effect of SPEYES technologies. The findings are confirmed by human-in-the-loop simulations.

I. Introduction

A. SPEYES System Overview

This paper discusses an agent-driven DDD simulation model to assess the SPEYES system. Historically, rapid post-conflict stability has been attained through high-troop densities. A key motivating theme leading to the SPEYES effort is the challenge of trying to enhance the SASO effectiveness with a limited number of forces through various force-multiplying technologies. The force multiplying technologies include: *Sensing* - low-cost, easily-emplaced, camouflaged sensors (video, acoustic, Infrared) to provide urban situation awareness; *Situation Awareness/Command and Control* (SA/C^2) - multi-echelon software tools tailored for distributed operations (planning, dynamic resource management, simulation, mission rehearsal); and *Shaping* - non-lethal, and explosives ordinance disposal (EOD) shaping tools, for example, to diffuse adversaries, crowds, and improved explosive devices) [1]. Figure 1 illustrates notionally how 'security' coverage could be expanded with the introduction of various technologies, e.g., tracer RFIDs, non-lethal weapons, smart scheduling, multi-spectral cameras, swarming unmanned vehicles.

The force-multiplying enablers increase the effectiveness of forces for SASO by means of better threat recognition, increased situational awareness and intelligence of the post-conflict environment, and improved decentralized operations. The SPEYES sensing technologies will increase the speed and ability of the forces to gather intelligence by the combined use and smart placement of visual, acoustic, spectral, and infrared sensors. The information gathered from the sensor observations allows troops and commanders to gain better situational awareness, and facilitates more efficient scheduling and patrolling, including the utilization of innovative shaping technologies [6]. Better and faster observations, combined with more

*This work was supported by the Information Exploitation Office of the Defense Advanced Research Projects Agency. This document carries the Distribution Statement "A" (Approved for Public Release, Distribution Unlimited).
[5]Correspondence: krishna@engr.uconn.edu

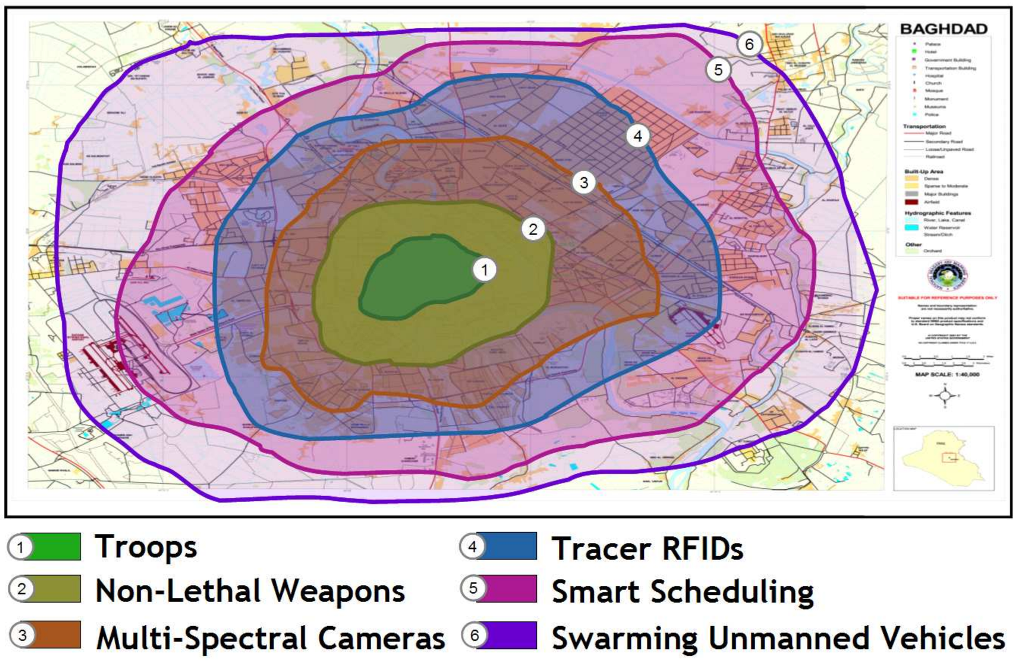


Fig. 1. The Notional Effect of Force Multipliers

accurate and timely orientation, permits better and more focused decisions leading to quicker and more effective actions on the part of the US forces.

B. Methodology

The challenges to evaluate SPEYES system include the determination of which security technologies create significant difference, how many of them are needed, and where they should be located to establish the desired force multiplying effects. We employed optimization-based dynamic planning and scheduling agents [2] within the Distributed Dynamic Decision-making (DDD) simulator [3] to quantify the force multiplying effect of SPEYES technologies.

The nature of the security and stability operations is similar in many respects to the emergency response service problems in urban operations research. The resource allocation, deployment, and incidence forecasting are fundamental issues in both. The Square Root Law [4] from urban operations research guided the agent-driven simulation-based efforts. The law is derived from a simple realization that distance is the square root of area. Thus, the traveled distance D, and consequently, the response-time T_r, of N_0 patrol units is proportional to the square root of the effective area A:

$$E[T_r] \approx \frac{E[T_r]}{v_c} = \frac{c}{v_c}\sqrt{\frac{A}{N_0(1-\rho)}} \tag{1}$$

where v_c denotes the patrol speed, ρ represents the busy (utilization) rate of the patrol unit, and c is a constant. What this law states is the following. Suppose we are able to reduce the response time from T_{r1} to T_{r2} with SPEYES technologies ($T_{r1} > T_{r2}$) for a given area A. Then, if we are willing to tolerate the same response time of T_{r1} with and without SPEYES technologies, then the same number of forces with SPEYES technologies can cover a larger area $A_2 = A\left(\frac{T_{r1}}{T_{r2}}\right)^2$, or equivalently, we can reduce the size of the forces to cover the same area.

We propose to utilize the lessons learned in urban operations research to show the improvements in timeliness, efficiency, and effectiveness stemming from the utilization of a SPEYES system. Operational efficiency was gleaned from the ability to maintain the same degree of force protection, while decreasing the number of personnel, amount of equipment, and cost and/or resources needed to sustain a particular level of protection. System effectiveness was gauged through improved force protection and incident

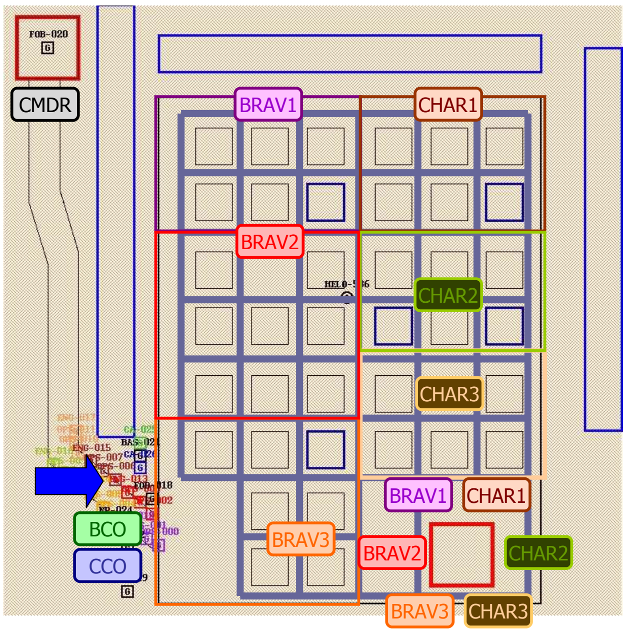


Fig. 2. The Simulation Scenario Map, Area of Responsibility, and Patrol Routes

prevention. Accordingly, keys to the agent-driven simulation for SPEYES system evaluation include: (i) systematic evaluation of the impact of SPEYES technologies; (ii) maximizing the effective number of patrols, i.e., troops; (iii) minimizing the response time to increase the area of coverage (for the same response time).

Accordingly, this paper is organized as follows. Section II-A presents the SASO scenario utilized for the simulation. A brief introduction to the DDD simulator is provided in Section II-B, and is followed by a discussion on the optimization-based agent in Section II-C. The outlines of performance measures and simulation results are presented in Sections III-A and III-B, respectively. The paper is concluded with a summary of findings and future research in Section IV.

II. Agent-driven Simulation

A. Scenario

The simulation utilized a military scenario similar to the one used at the National Training Center in Ft. Irwin, California, where a simulated Tiefort City (TC), serves as a setting for urban warfare training. The simplified geographical map of TC is shown in Figure 2. In this set-up, US and coalition forces are conducting multi-phase operations to maintain security and stability in TC. The forces include 1 Battalion and 4 Companies. The ALPHA and DELTA companies are tasked to reconnaissance operations; CHARLIE and BRAVO companies are assigned to cordon, search, and secure operations in TC. The latter two companies have moved from their position at the Forward Operating Base (FOB) to the entrance of TC, and are ready to commence operations within their areas of responsibility, as outlined in Figure 2, to secure the power plant in TC and suppress an ongoing insurgency.

The command organization and force composition of the two companies (CHARLIE and BRAVO), that conduct the SASO operations, is shown in Figure 3. It shows assets from company-, as well as, battalion-levels. Each company is comprised of three platoons controlled by individual commanders. Each platoon is comprised of two operations (OPS) units (a joint configuration of a Tank and a Bradley), an engineering squad (ENGR), four mobile fire teams (MFTs), and two anti-sniper teams (AST). The OPS unit carries two mobile fire and anti sniper teams on a Bradley vehicle, and dispatches them to conduct operations inside the city. At the battalion level, there is an explosive ordinance disposal (EOD) squad, three military police (MP) squads dealing with detainees and conducting interrogations, two helicopters (HELO) conducting fire support, three medical squads (MED), two unmanned aerial vehicles (UAVs), two allied police (POL) squads, and a Q-36 radar. Details of asset resources and attributes can be found in [1].

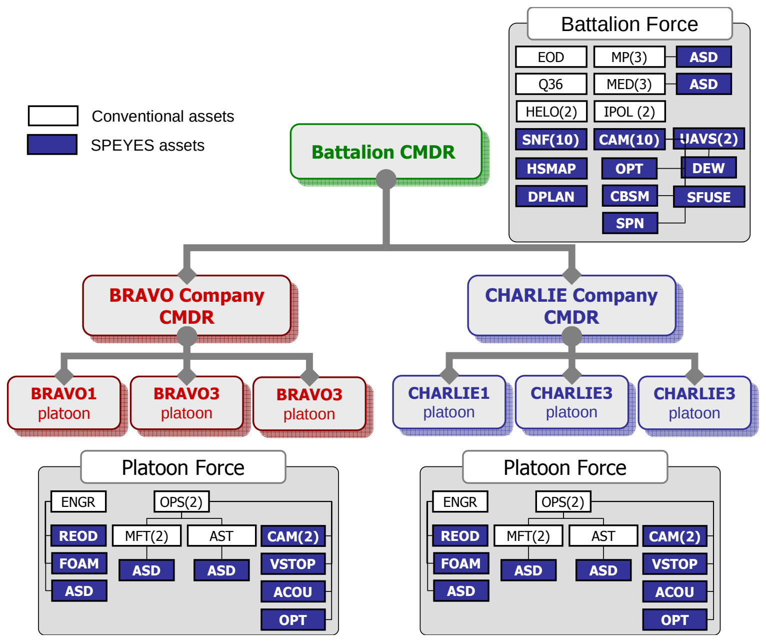


Fig. 3. The Simulation Organizational Structure

The objectives of CHARLIE and BRAVO companies are to stop the violence, establish control over the city, secure critical infrastructure, and achieve a return to normalcy in which routine civil and economic activities can take place without disruption. To accomplish these goals, friendly forces must search buildings to locate and secure weapons caches, interrogate suspected insurgents, search and neutralize snipers and foreign fighters, conduct surveillance, clear and secure strategic buildings, avoid IEDs, prevent VBIED attacks, secure the power plant, establish check points, and deal with crowds in the city.

In this effort, we modeled various force multiplying technologies as a single integrated SPEYES system. To assess the performance effects of the various SPEYES technologies in terms of force multiplier payoffs, we simulated and then compared three organizational configurations for the forces: (i) the organization outlined in Figure 3 without SPEYES technologies, (ii) the organization outlined in Figure 3 with SPEYES technologies, and (iii) an organization half the size of that in Figure 3 with SPEYES technologies.

B. The DDD Simulator

To realize the aforementioned assessment objectives, we utilized a flexible, controllable, distributed simulation paradigm suitable for examining the security dynamics in a post-conflict mission environment, and for quantifying the potential performance improvements of a SPEYES system over current practices. Such a simulation paradigm is the distributed dynamic decision-making (DDD) simulator [3], which allows researchers to examine the interactions between the mission structure, and the way in which the organization tasked to execute the mission is structured. The DDD simulator allows for manipulations of organizational structures, such as authority, information, communication, resource ownership, task assignment, etc., as well as mission and environmental structures, representing a variety of task classes, and assets (platforms with sub-platforms, sensors, and weapons) with specified resource capabilities. Model-driven experiments with the DDD simulator utilize four major modeling entities: assets, tasks, resources, and decision makers (commanders).

Tasks are activities/events that require certain resources for successful execution with appropriate friendly assets. Tasks are generated from a mission plan or event list, and can represent individual targets or enemy actions, location of the search to be conducted, rescue operations, defense and attack, monitor, etc. A mission task T_i (or simply i) is characterized by the following basic attributes: (i) task arrival time, a_i (time at which the task appears in the mission scenario); (ii) task processing time window, t_{a_i} (maximal time available from the start of task execution to its finish; this time window is used to synchronize assets assigned to this task; (iii) estimated task processing time, t_i (the interval between the time the first asset starts executing the task and the time at which the last asset finishes the task; the task processing time must not exceed the task processing time window, $t_i \leq t_{a_i}$); (iv) accuracy of task processing, α_i, which depends on how well the resource requirements of a task are matched by the resources allocated to the task; (v) task start time, s_i (the time that a task execution is started, which is obtained during mission execution); (vi) geographical location of task, (x_i, y_i); (vii) resource requirement vector $[r_{i1}, r_{i2}, ..., r_{iL}]$, were r_{ij} is the number of units of resource of type l required for successful processing of task i (l =1,. . . ,L, where L is the number of resource types); this vector defines the resources required to successfully process (attack) the task; and (viii) task value, ω_i (task value reflects the fact that not all tasks are equally important). In addition, there exists a dependency diagram that details the interrelationships among tasks: (i) task precedence; (ii) inter-task information flow; and (iii) input-output relationships between tasks, i.e., the *task graph*. This directed acyclic task-precedence graph represents a plan to execute the mission.

A platform (asset) is a controllable and/or movable unit, which represents a physical entity of an organization to process a task, such as individual weapons or weapon systems, sensors, or human teams at any level of granularity (squad, platoon, company, etc.). Resources represent the capabilities of an asset (asset-resource capabilities), and tasks require resources to process (task-resource requirements). Thus, in order to process a task, the organization needs to match the asset capabilities to the task requirements. Each platform P_m (or simply m) $(m = 1, \ldots, K)$ has several attributes that uniquely define this platform: (i) sub-platforms, i.e., additional assets that reside on-board a parent platform that only become active after being 'launched' from the parent; (ii) ownership, i.e., only owners of the platforms are able to move, carry a pursuit or attack with them, or launch sub-platforms; owners of parent platforms are not necessarily owners of the sub-platforms; (iii) sensors, i.e., specify effectiveness ranges for task detection, measurement, and identification or classification; (iv) geographical location, i.e., (x_m, y_m); (v) maximum velocity v_m, that defines how fast a platform can travel; and (vi) resource capability vector $[\hat{r}_{m1}, \hat{r}_{m2}, ..., \hat{r}_{mL}]$, where $\hat{r}_{ml}$ specifies the number of units of resource type l available on platform m. A platform can be used to attack any task, but the ranges of platform's weapons depend on the task type or class. Assets must be routed among task locations to execute the assigned tasks. Assets can begin to process the same task at different times, but must be synchronized to complete a task in a specified task processing window.

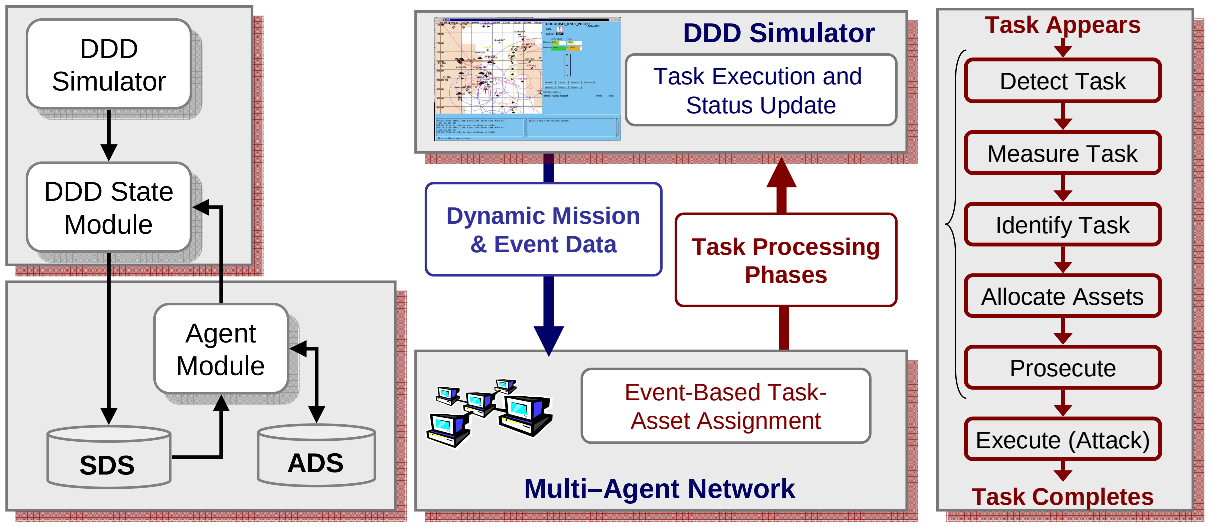
A decision-maker (DM), DM_j (or simply j), is an entity with information-processing, decision-making, and operational capabilities that can control the necessary resources to execute mission tasks, provided that it does not violate the concomitant capability thresholds. In the current simulation context, the DMs are commanders who have the ownership/control over their assigned assets, and make decisions about which tasks to select for execution and with which assets.

C. Agent Model

i. Overview

The optimization-based dynamic task processing agent framework for the DDD simulator [2] was adapted for the current study by adding features relevant to a SPEYES security system. These included patrol routes and smart dynamic assignment of SPEYES assets. The architecture of the agent-driven simulation framework is shown in Figure 4.a and b. The Shared Data Storage (SDS) provides the DDD state variables (estimates of own and adversary states) that serve as agents' situation awareness (SA). The Agent Data Storage (ADS) holds agents' normative behaviors that are specific to a given environment (e.g., geographical responsibility for each decision-maker, possible patrol routes), the various mission monitoring, and (re)planning data and measures.

In the DDD environment, each DM is equipped with a set of assets that determine the DM's resource capability, as well as SA. In this environment, SA represents the ability to monitor task and/or asset status, to detect existence of tasks, measure characteristics of tasks, and classify tasks. Situational awareness of a DM is limited by the constraints of his assets. A DM may not be aware of the existence of a task, if the task is outside his asset coverage. Such knowledge has to be communicated among DMs within a team (among coordinating partners) to enhance the team's collective SA. The current implementation addresses this communication issue as additional delays among DMs, which results in delays in task completion.



(a) Core Architecture (b) DDD-Agent Interaction Schema (c) Task Processing Phases

Fig. 4. Agent-driven Simulation within DDD Paradigm

In the SPEYES security system, the agents, which represented a set of cooperative DMs, employed *centralized asset-task assignment*, and *decentralized task execution*, taking into account communication and coordination delays across the team hierarchy (see Figure 5.a). The execution of a task can only begin when all of its predecessors are completed ('READY'), and all assets from the assigned asset group arrive at the appropriate locations. In addition, an asset can only process one task at a time. The objective of task-asset assignment is to distribute the available assets among tasks, such that the overall mission completion time is minimized.

To process tasks, agents follow a multi-stage process shown in Figure 4.c. When a task appears in the scenario, it should first be detected with appropriate sensors. Second, the task must be measured to determine its attributes, and must be identified to determine its class. When the task is selected for execution, the assets that will prosecute this task must be allocated. This is done according to task-resource requirements, asset-resource capabilities, task and asset attributes, current utilization of assets (location, assignment), and 'attack' and 'be attacked' ranges of the assets. The procedures for the centralized task-asset assignment are outlined in Figure 5.b, wherein a set of selected 'READY' tasks are allocated to groups of assets such that the aggregated resource capabilities meet or exceed the demands of the task. The decentralized task execution involves coordinated launching of assets (subplatforms, weapons) from their parent assets (platforms), movement of assets from their current locations to the task's location, and executing the task within a task-specific window of opportunity. The selected assets are moved by DM owners using appropriate commands/controls into position within the 'attack' range of the task. When assets come into position to execute the task, they start the attack phase, which takes the time specified in the 'duration' attribute of the task.

ii. Formal Task Processing Procedures

The agent task processing procedures, outlined in Figure 5.b, adopt the multi-dimensional dynamic list scheduling method (MDLS) [5], wherein it finds the platform-task allocation and mission schedule by sequentially assigning tasks to platforms until the task set is exhausted. MDLS heuristic has two main steps: (1) select the task to be processed; and (2) select the group of platforms to be assigned to it for processing.

The priority of a task i, $P(i)$, is influenced by the current assignment information and precedence structure, such as the opportunity window to process task i, t_{a_i}; the value of task i, ω_i; and other tunable criteria, $f_c(i)$:

$$P(i) = f_c(i)\frac{\omega_i}{1+t_{a_i}} \tag{2}$$

A group of platforms, $GROUP$, is selected to process the selected task such that the task's resource

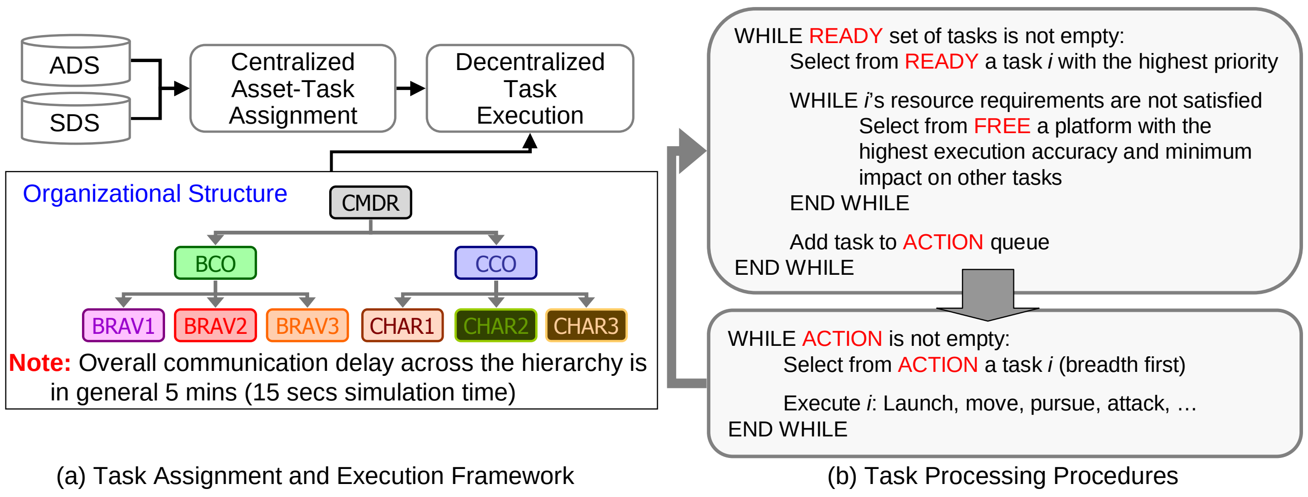

ADS
SDS
Centralized Asset-Task Assignment
Decentralized Task Execution
Organizational Structure
CMDR
BCO
CCO
BRAV1
BRAV2
BRAV3
CHAR1
CHAR2
CHAR3
Note: Overall communication delay across the hierarchy is in general 5 mins (15 secs simulation time)
(a) Task Assignment and Execution Framework
WHILE READY set of tasks is not empty:
Select from READY a task i with the highest priority
WHILE i's resource requirements are not satisfied
Select from FREE a platform with the highest execution accuracy and minimum impact on other tasks
END WHILE
Add task to ACTION queue
END WHILE
WHILE ACTION is not empty:
Select from ACTION a task i (breadth first)
Execute i: Launch, move, pursue, attack, …
END WHILE
(b) Task Processing Procedures

Fig. 5. Task Assignment and Processing Framework and Procedures

```
Find the set:
    $READY1 = \left\{ i \in READY | \sum_{m \in FREE} r_{ml} \geq R_{il}, \forall l = 1, ..., L \right\}$
DO UNTIL $READY1 = \emptyset$
    Task selection:
        $i^* = \arg \max_{i \in READY1} P(i)$
        $READY1 \leftarrow READY1 \backslash \{i^*\}$
    Platform group selection for $i = i^*$:
    Find the set:
        $FREE1 = \left\{ m \in FREE | \sum_{l=1}^{L} \min(r_{ml}, R_{il}) \neq 0 \right\}$
        $v_{im} = \pi_{im}, \forall m \in FREE1$
        $GROUP = \emptyset$
        DO UNTIL $\sum_{m \in GROUP} r_{ml} \geq R_{il}, \forall l = 1, ..., L$
            $m^* = \arg \max_{m \in FREE1} v_{im}$
            $FREE1 \leftarrow FREE1 \backslash \{m^*\}$
            $GROUP \leftarrow GROUP \cup \{m^*\}$
        END DO
END DO
```

Fig. 6. Task and Platform Group Selection Procedure

requirement vector is component-wise less than or equal to the aggregate resource capability vector of the group of platforms assigned to it:

$$\sum_{P_m \in GROUP} \hat{r}_{ml} = \sum_{m=1}^{n_P} y_{mi} \hat{r}_{ml} \geq r_{il}, \forall l = 1, ..., L \tag{3}$$

The notation y_{mi} specifies the asset-task assignment such that

$$y_{mi} = \begin{cases} 1, \text{ if asset } P_m \text{ is assigned to task } T_i \\ 0, \text{ otherwise} \end{cases} \tag{4}$$

The DDD simulator requires concurrent task execution within a small time window by all assigned platforms. Since the travel times of the assigned platforms differ, the closer or faster platforms should wait for others before attacking the task or, alternately, all assigned platforms should synchronize their departure times so as to arrive at the task location concurrently, viz., begin task execution together. The goal of the platform group selection is to optimize the overall mission completion accuracy, as well as the overall mission completion time. Accordingly, the task-asset assignment aims to balance the trade-off between minimizing the completion time and maximizing the accuracy, while avoiding the allocation of assets needed by other tasks. The first goal is operationalized by minimizing the travel time of platform m to task i, τ_{mi}. The second consideration is realized by maximizing the ratio of accuracy when assigning platform m to task i compared to assigning it to any other task $j \neq i$. Accordingly, the cost of assigning platform m to task i, π_{im}, is calculated as:

$$\pi_{im} = \tau_{mi} \frac{\frac{\omega_i+1}{\omega_i}\alpha_i}{\sum\limits_{j \in READY1 \backslash \{i\}} \frac{\omega_j+1}{\omega_j}\alpha_j} \tag{5}$$

Accuracy of a task i assigned to platform m depends on task-resource requirements, platform-resource capability, and task value, and is computed as follows:

$$\alpha_i = \frac{100 \times \omega_i}{n_{R_i}} \sum_{l=1}^{L} \left(\frac{\min\left(\hat{r}_{ml}, r_{il}\right)}{r_{il}} \right)^2 \tag{6}$$

Note that n_{R_i} represents the number of resource types that a task i requires.

One of the key improvements to the agent-driven DDD simulation in realistically depicting the relevant aspects of a SPEYES system is the addition of a smart scheduling technique to enhance patrol effectiveness [6]. This technique combines sensing and SA/C^2 technologies. The sensing technologies include various sensors to provide persistent surveillance, which are able to detect, identify, and track targets and identify threats using ISR platforms operating at standoff distances, especially in urban areas in the presence of obstructions such as buildings, canopies, foliage, camouflage, etc. The SA/C^2 technologies incorporate a dynamic scheduling technique, which utilizes the persistent ISR data to select the shortest and safest patrol routes for asset maneuvers.

Specifically, the smart scheduling takes into account specified routes, which depicts roads in TC, as shown in Figure 2. That is, the travel time of platform m to task i, τ_{im}, is computed as the aggregate time to travel the shortest distance following the safest connected segments of TC roads starting from location m to location i, i.e., from (x_m, y_m) to (x_i, y_i):

$$\tau_{mi} = \frac{1}{v_m} \min_{SEGMENT} \left(d_{m0} + \sum_{k \in SEGMENT1} \left(d_{(k-1)k} \right) + d_{n_S i} \right) \tag{7}$$

Here, d_{m0}, $d_{(k-1)k}$, and $d_{n_S i}$ represent the distance traveled from (x_m, y_m) to the first road segment, from the current segment to the next, and from the last segment to the task location (x_i, y_i), respectively. The notations $SEGMENT$ and $SEGMENT1 \subset SEGMENT$ denote the set of available and selected TC road segments, respectively; n_S represents the cardinality of $SEGMENT1$; and v_m denotes the platform's maximum velocity. Note that due to constant presence of threats, whose data are accessible through persistent surveillance, not all TC roads are safe to travel; unsafe road segments are deemed unavailable, i.e., excluded from $SEGMENT1$. Accordingly, we have the platform group selection procedure shown in Fig. 6.

III. Simulation Results

A. Performance Measures

The force multiplier payoff for the technologies in the integrated SPEYES system was assessed in terms of timeliness (e.g., throughput, latency, response time), efficiency (e.g., troops efficiency, assets damaged, casualties), and effectiveness (secured area as a function of patrol units and incident rates with and without the SPEYES technologies, task completion rate, failed tasks, detection coverage) measures. The following is a summary of representative measures.

In our simulations, we have considered throughput measures for different task classes for both task processing and identification (e.g., sensor detection throughput, which corresponds to the rate of detecting IEDs, RPGs, and snipers). The overall task processing throughput, which is the ratio of the number of tasks processed to the mission completion time, indicates the rate at which activities are accomplished by the friendly forces:

$$\text{Overall Processing Throughput} = \frac{\sum\limits_{k=1}^{n_K}\sum\limits_{i=1}^{n_T}\sigma_{ik}}{\max\limits_{i}\left(0, s_i + t_i\right)} \tag{8}$$

Here, n_K and n_T denote the number of task classes and the number of tasks, respectively. The denominator of the above equation is the mission completion time, which indicates the speed with which the mission was completed. The variables s_i and t_i represent the start and processing time of task i, respectively; and

$$\sigma_{ik} = \begin{cases} 1, \text{ if task } i \text{ of class } k \text{ has been completed} \\ 0, \text{ otherwise} \end{cases} \tag{9}$$

Similarly, the overall task detection throughput can be computed as follows:

$$\text{Overall Detection Throughput} = \frac{\sum\limits_{k=1}^{n_K}\sum\limits_{i=1}^{n_T}\delta_{ik}}{\max\limits_{i}\left(0, d_i\right)} \tag{10}$$

The variable d_i denotes the detection time of task i; and

$$\delta_{ik} = \begin{cases} 1, \text{ if task } i \text{ of class } k \text{ has been detected} \\ 0, \text{ otherwise} \end{cases} \tag{11}$$

One of the important indicators of how efficiently the friendly forces can prevent and suppress enemy activities was the number of enemy attacks, which included attacks by IEDs, RPGs, snipers, and enemy fighters. The impact of enemy attacks was assessed in terms of an overall casualty measure; this was calculated based on the probability of friendly casualties due to a specific enemy attack. The probability of casualty from a specific task class is denoted by p_k. In the simulations, we assumed that p_k of an IED attack and a sniper or RPG attack is 50% and 20%, respectively.

$$\text{Total Casualties} = \sum_{k=1}^{n_K}\sum_{i=1}^{n_T}\sum_{m=1}^{n_P}\left(\kappa_{ik} y_{im} \nu_m \chi_m p_k\right) \tag{12}$$

Here, ν_m denotes the number of troops belonging to asset P_m; y_{im} specifies the asset-task assignment as defined in equ. (4); and

$$\kappa_{ik} = \begin{cases} 1, \text{ if task } T_i \text{ belongs to task class } k \\ 0, \text{ otherwise} \end{cases} \tag{13}$$

The notation χ_m indicates whether asset P_m is damaged

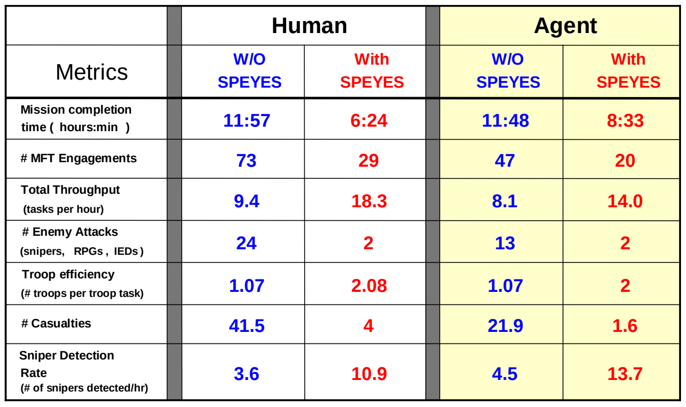
$$\chi_m = \begin{cases} 1, \text{ if asset } P_m \text{ is damaged} \\ 0, \text{ otherwise} \end{cases} \tag{14}$$

The troop efficiency signifies the average number of troops that are available to execute a single troop task. The number of troop tasks is computed as the tasks that require troops and that cannot be executed by other assets.

$$\text{Troop Efficiency} = \frac{\sum\limits_{m=1}^{n_P}\nu_m}{\sum\limits_{k=1}^{n_K}\sum\limits_{i=1}^{n_T}\lambda_{ik}} \tag{15}$$

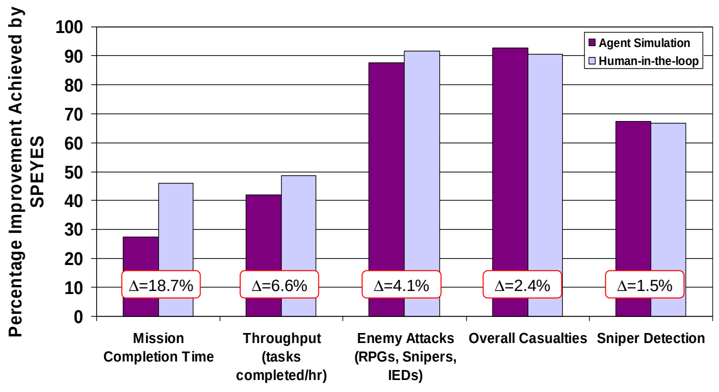
where

$$\lambda_{ik} = \begin{cases} 1, \text{ if task } i \text{ of class } k \text{ requires troops} \\ 0, \text{ otherwise} \end{cases} \tag{16}$$



Metrics	Human: W/O SPEYES	Human: With SPEYES	Agent: W/O SPEYES	Agent: With SPEYES
Mission completion time (hours:min)	11:57	6:24	11:48	8:33
# MFT Engagements	73	29	47	20
Total Throughput (tasks per hour)	9.4	18.3	8.1	14.0
# Enemy Attacks (snipers, RPGs, IEDs)	24	2	13	2
Troop efficiency (# troops per troop task)	1.07	2.08	1.07	2
# Casualties	41.5	4	21.9	1.6
Sniper Detection Rate (# of snipers detected/hr)	3.6	10.9	4.5	13.7

(a) One-to-one Comparison of Human and Agent Simulations



(b) Comparison of Improvement Obtained with SPEYES

Fig. 7. The Agent-Human Result Validation

Another efficiency measure is the number of engagements by specific assets with troops (e.g., number of engagements by MFT teams had a significant impact on speed of operations). The number of engagements of an asset with troops can be computed as follows:

$$\text{Number of Engagements by Asset } P_m = \sum_{k=1}^{n_K} \sum_{i=1}^{n_T} y_{im} \lambda_{ik} \tag{17}$$

The organizational effectiveness is gauged by comparing the secured area, as a function of patrol units and incident rates, of the aforementioned organizational configurations, i.e., without SPEYES technologies, with SPEYES technologies, and an organization half the original size with SPEYES technologies. The number of enemy attacks, overall troop casualties, and throughput are representatives of measures to asses the increase in secured area due to SPEYES technologies.

B. *Results and Discussion*

A series of simulations were conducted to calibrate the agent performance with that achieved in human-in-the-loop simulations. The differences between the agent and human task processing is largely attributed to pre-planning and partial domain knowledge available to humans, but not to agents. The similarities and differences between the performances of agent simulations and human-in-the-loop runs can be seen in Figure 7.a. Despite the differences, nevertheless, the trends exhibited by human and agent simulations were comparable, i.e., $1.5\% \leq \Delta \leq 18.7\%$, (see Figure 7.b) and indicated consistent benefits accrued by a SPEYES system.

To assess the impact of the SPEYES system, we conducted agent-based DDD simulations. Three aforementioned TC mission scenarios (without SPEYES system, with SPEYES, and with SPEYES under a 50% troop reduction) were considered. These runs addressed the improvements in timeliness, efficiency, and effectiveness achievable with the SPEYES system.

The results, shown in Figure 8, demonstrated substantial performance improvements due to the security technologies: (i) a significant improvement in task processing speed, which is indicated by an increase of 42% in throughput and a reduction of 22% in mission completion time; (ii) a substantial improvement in task processing efficiency, which is shown by a reduction of 85% and 93% in the numbers of enemy attacks and overall casualties, respectively; and (iii) a sizable improvement in task effectiveness, which is indicated by 47% improvement in troop efficiency, and an increase of 67% in the sniper detection coverage. Furthermore, we found that even with the 50%-reduced force (3 platoons instead of 6 platoons), a force with the SPEYES system was able to achieve a significant improvement in performance over regular operations with a full force without SPEYES, thus confirming the force multiplier effects achievable by utilizing the SPEYES system. These results supported similar evidence observed in human-in-the-loop simulations in [1].

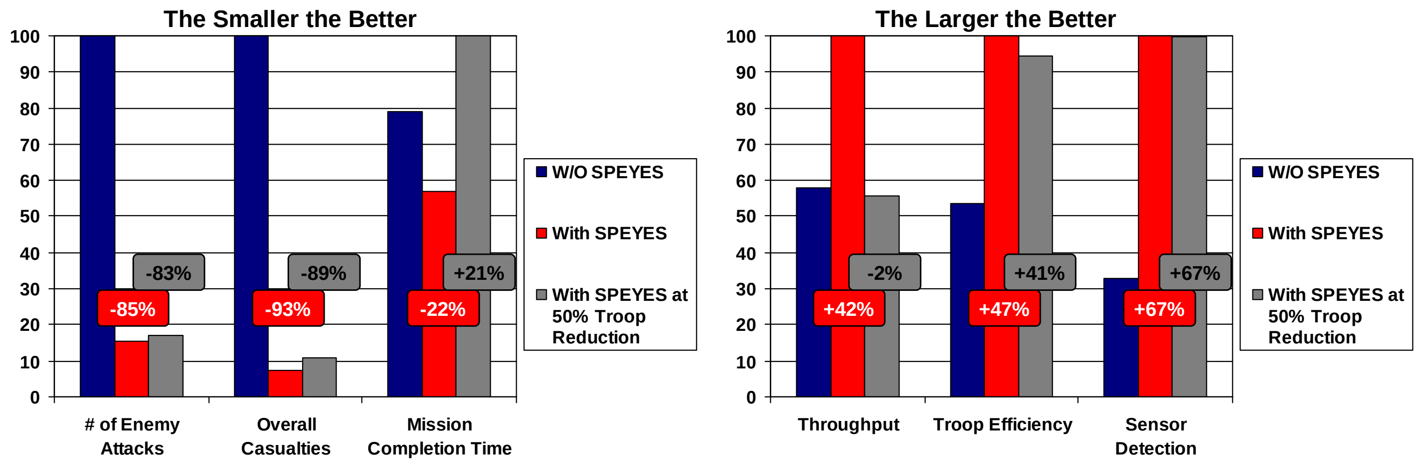


Fig. 8. Normalized Performance and Process Measures (Δ = improvement obtained with SPEYES system compared to without SPEYES case)

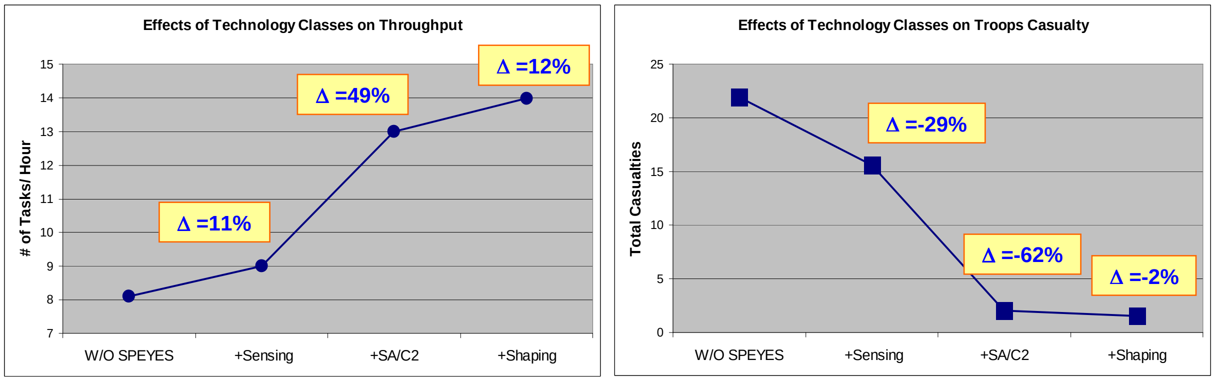


Fig. 9. SPEYES Component Technology Benefits (Δ = relative improvement compared to W/O SPEYES case)

Next, we conducted sensitivity analysis to evaluate the impact of specific classes of SPEYES technologies. In particular, we wanted to analyze the impact of sensing technologies; SA/C^2 technologies; and shaping technologies. Since the DDD simulations did not explicitly model SA/C^2 functionality (e.g., information fusion, integrated threat maps, entity and behavioral threat models, etc.), the performance contributions of the SA/C^2 technologies were assumed to be embedded within the sensing and shaping technologies and contributed two-third of sensing and shaping performance improvements. Figure 9 displays the relative performance improvements of SPEYES component technologies. The sensing technologies increased throughput by 11% and decreased casualties by 29%; the SA/C^2 technologies improved throughput by 49% and reduced casualties by 62%; and the shaping technologies enhanced throughput by 12% and decreased casualties by 2%.

These results are intuitive and underscore the importance of the system integration component of SPEYES, viz., combined sensing, SA/C^2, and shaping technologies. The effect of the combined technologies was threefold. First, the soldiers obtained a better situational awareness about the threats that should be avoided while on patrol and conducting regular security operations. Second, the information supplied the troops with better and faster intelligence regarding problem areas, which enabled them to react quickly and precisely; thus augmenting the force capability to respond to broader threat variety and decreasing casualties. Finally, the combined SPEYES technologies provided threat deterrence, i.e., reduced the total number of hostilities throughout the mission.

IV. Conclusion

The focus of this paper was to present agent-driven DDD simulation model to evaluate a SPEYES system in terms of timeliness, effectiveness, and efficiency of SASO operations on US and coalition forces.

The behaviors of the optimization-based DDD agents were calibrated to those of human-in-the-loop DDD simulations. The agent-driven simulation results indicated that the integrated SPEYES technologies, which included sensing, SA/C^2, and shaping technologies, provided significant performance improvements to the force across all measures. Even at 50%-reduced force, the SPEYES system maintained significant performance improvements over regular operations with a full force and without a SPEYES system, thus confirming the force multiplier effect introduced by utilizing SPEYES. The findings are in accordance to the ones observed in human-in-the-loop simulation results presented in [1].

Acknowledgements
The authors would like to thank and acknowledge Daniel Serfaty, Keith Baker, Adam Bailey, Paul Titus, Stacy Lovell, Gabriel Ganberg, Woody Spring, Fred Flynn and David Hurley, and the rest of the SPEYES team for their support in improving the DDD simulator, developing performance and process measures, and building SPEYES DDD scenarios.

References

[1] Popp, R. , G. M. Levchuk, D. Serfaty, D. Allen, C. Meirina, F. Yu, S. Ruan, K. R. Pattipati, and M. Lazaroff. (2005). ”SPEYES: Sensing and Patrolling Enablers Yielding Effective SASO,” *Proceedings of the 2005 IEEE Aerospace Conference*, Big Sky, MT, March 2005.

[2] Meirina, C., G. M. Levchuk, and K.R. Pattipati. (2003). ”A Multi-Agent Decision Framework for DDD-III Environment,” *2002 International Command and Control Research and Technology Symposium*, Monterey, CA, June 2003, 21 pages.

[3] Kleinman, D. L., P. Young, and G. S. Higgins. (1996). ”The DDD-III: A Tool For Empirical research in Adaptive Organizations”, *Proceedings of the 1996 Command and Control Research and Technology Symposium*, Monterey, CA.

[4] Kolesar, P. and Blum, E.H. (1973). ”Square Root Law for Fire Engine Response Distances,” *Management Science*, Vol. 19, No. 12, pp. 1368-1378.

[5] Levchuk, G.M., Levchuk, Y.N., Luo Jie, Pattipati K.R., and Kleinman D.L. (2002). ”Normative design of organizations-part I: Mission planning,” *IEEE Trans. on Systems, Man, and Cybernetics-Part A*, 32(3):346-359.

[6] Ruan, Sui, C. Meirina, F. Yu, and K. R. Pattipati. (2005). ”Patrolling in Stochastic Environment,” Submitted to the *2005 International Command and Control Research and Technology Symposium*, McLean, VA, June 2005, 10 pages..