2014-07-31

Implementation of Monte Carlo Tree Search (MCTS) Algorithm in COMBATXXI using JDAFS

Teter, Michael; Buss, Arnold; Darken, Christian; Baez, Ricardo

TRADOC Analysis Center, "Implementation of Monte Carlo Tree Search (MCTS) Algorithm in COMBATXXI using JDAFS" (2014) TRAC-M-TR-14-031, pp. B2-B9.
http://hdl.handle.net/10945/70988

TRAC-M-TR-14-031
31 July 2014

Implementation of Monte Carlo Tree Search (MCTS) Algorithm in COMBATXXI using JDAFS

TRADOC Analysis Center
700 Dyer Road
Monterey, California
93943-0692

This page intentionally left blank.

TRAC-M-TR-14-031
31 July 2014

Implementation of Monte Carlo Tree Search (MCTS) Algorithm in COMBATXXI using JDAFS

MAJ Michael Teter
Dr. Arnold Buss
Dr. Christian Darken
LTC Ricardo Baez

TRADOC Analysis Center
700 Dyer Road
Monterey, California
93943-0692

This study cost the
Department of Defense approximately
$37,400 expended by TRAC in
Fiscal Year 14.
Prepared on 20140819
TRAC Project Code # 060025.

This page intentionally left blank.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
31-07-2014	Technical Report	Jul 13 - Jul 14

4. TITLE AND SUBTITLE

Implementation of Monte Carlo Tree Search (MCTS) Algorithm in COMBATXXI using JDAFS

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

MAJ Michael Teter
Dr. Arnold Buss
Dr. Christian Darken
LTC Ricardo Baez

5d. PROJECT NUMBER

060025

5e. TASK NUMBER

5f. WORK UNIT NUMBER

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

TRADOC Analysis Center - Monterey
700 Dyer Road
Monterey CA 93943

8. PERFORMING ORGANIZATION REPORT NUMBER

TRAC-M-TR-14-031

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

TRADOC Analysis Center - Methods and Research Office
255 Sedgewick Rd
Ft Leavenworth KS

10. SPONSOR/MONITOR'S ACRONYM(S)

TRAC-MRO

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

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT: Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The implementation of the Monte Carlo Tree Search (MCTS) algorithm into the Combined Arms Analysis Tool for the 21st Century (COMBATXXI) project is an extension of work completed in FY13. The TRADOC Analysis Center - Methods and Research Office (TRAC-MRO) sponsored this iteration in an attempt to test the feasibility implementing the algorithm into the COMBATXXI simulation environment. For further details on the specific algorithm of more background information see appendix and the previous technical report.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER** *(include area code)*

This page intentionally left blank.

Table of Contents

List of Figures

This page intentionally left blank.

Implementation of Monte Carlo Tree Search (MCTS) Algorithm in COMBATXXI using JDAFS

Chapter 1
Introduction

Background

The implementation of the Monte Carlo Tree Search (MCTS) algorithm into the Combined Arms Analysis Tool for the 21st Century (COMBATXXI) project is an extension of work completed in FY13.[1] The TRADOC Analysis Center - Methods and Research Office (TRAC-MRO) sponsored this iteration in an attempt to test the feasibility implementing the algorithm into the COMBATXXI simulation environment. For further details on the specific algorithm of more background information see appendix B and the previous technical report.[2]

Problem Statement

To execute the Monte Carlo Tree Search (MCTS) algorithm for autonomous decision-making agents within COMBATXXI.

Issue 1: Integration of algorithm using JDAFS.

EEA 1.1: Will the algorithm execute using JDAFS to build state space?

EEA 1.2: What changes to the algorithm are necessary for integration into COMBATXXI?

Issue 2: COMBATXXI Interface.

EEA 2.1: Does the algorithm interface with COMBATXXI?

EEA 2.2: How does the algorithm change the force on force decisions in COMBATXXI?

[1] See MAJ Christopher Marks et al. *Mission Command Analysis Using Monte Carlo Tree Search.* Tech. rep. TRAC-M-TR-13-050. 700 Dyer Road Monterey, California 93943: TRADOC Analysis Center - Monterey, 2013. URL: "https://ako.hq.tradoc.army.mil/sites/trac/MTRY/SitePages/Home.aspx".

[2] See ibid.

Constraints, Limitations, & Assumptions

- Constraints[3]
  - Complete by 30 JUN 14.
- Limitations[4]
  - We will carry out all experimentation in simulation environments for which we can obtain programmer support.
  - Rewards functions will be defined by the study team.
- Assumptions[5]
  - Implementation will not require modification of COMBATXXI.
  - Algorithm will operate outside of COMBATXXI.
  - Will require link between COMBATXXI and the algorithm.
  - Algorithm will use JDAFS to populate state-space.
  - Contractor support for coding will be available.

[3]Constraints limit the project team's options to conduct the research.
[4]Limitations are a project team's inabilities to investigate issues within the sponsor's bounds.
[5]Assumptions are research-specific statements that are taken as true in the absence of facts.

Chapter 2
Analysis and Methodology

In this chapter, we examine the methodology in which we approach the problem. Our methodology included four steps:

1. Define the Problem.
    - Literature review.
    - Assemble the team.
2. Scenario Development.
    - Create prototype use-case in JDAFS.
    - Expand use-case to implement using JDAFS.
3. Test and Evaluation.
    - Integrate the use-case in COMBAT XXI.
    - Test use-case against base-case.
4. Analysis
    - Documentation of implementation.

All steps which were completed are expanded in the following sections. After further exploration (See Project Termination Section) , this methodology was abbreviated and half of step 2 and all of step 3 were not completed.

Define the Problem

We conducted the literature review and assembled the team during the initial phase of the project. The literature review consisted of the materials referenced in the previous technical report[6] along with additional materials[7] to become familiar with the algorithm.

[6] See Marks et al., *Mission Command Analysis Using Monte Carlo Tree Search*, op. cit.

[7] see Mark HM Winands, Yngvi Björnsson, and Jahn-Takeshi Saito. "Monte-carlo tree search solver". In: *Computers and Games*. Springer, 2008, pp. 25–36.

Figure 1. COMBATXXI MCTS test scenario.

The project team initially held meetings with TRAC-White Sands Missile Range (TRAC-WSMR)[8] to understand the stakeholders concerns. We also met with the sponsor[9] for approval of the restated problem statement and technical approach.

Scenario Development

Prototype

The scenario development began with the scenario suggested by the previous tech report.[10] This is a brief overview as described in that report.

> "Our scenario consists of a group of "red" riflemen are advancing on a single "blue" grenadier in a fixed position (see figure 1). Armed with an assault rifle (60 rounds) and a grenade launcher (12 grenades), the grenadier must decide with which weapon to engage the enemy. By varying the number of red troops advancing on the blue grenadier, we can use the results of MCTS runs to gain insight into what situations the grenadier should choose each weapon over the other in order to get the best effects, and when it might be best to switch weapons during engagements. The red elements will execute COMBATXXI default behaviors, i.e., they will engage the blue entity once they it is within range. For the purpose of this analysis, the advancing red troops will remain relatively concentrated. The results of this analysis might be useful in informing the CombatXXI weapons selection behavior, which currently simply selects weapons from a weapon-priority list."[11]

[8]Mr. Blane Wilson of Mission Command and Mr. Dave Ohman of Modeling and Simulation
[9]Mr. Paul Works of the Methods and Research Office (MRO)
[10]See Marks et al., *Mission Command Analysis Using Monte Carlo Tree Search*, op. cit., pp. 54-56.
[11]See ibid., p. 54.

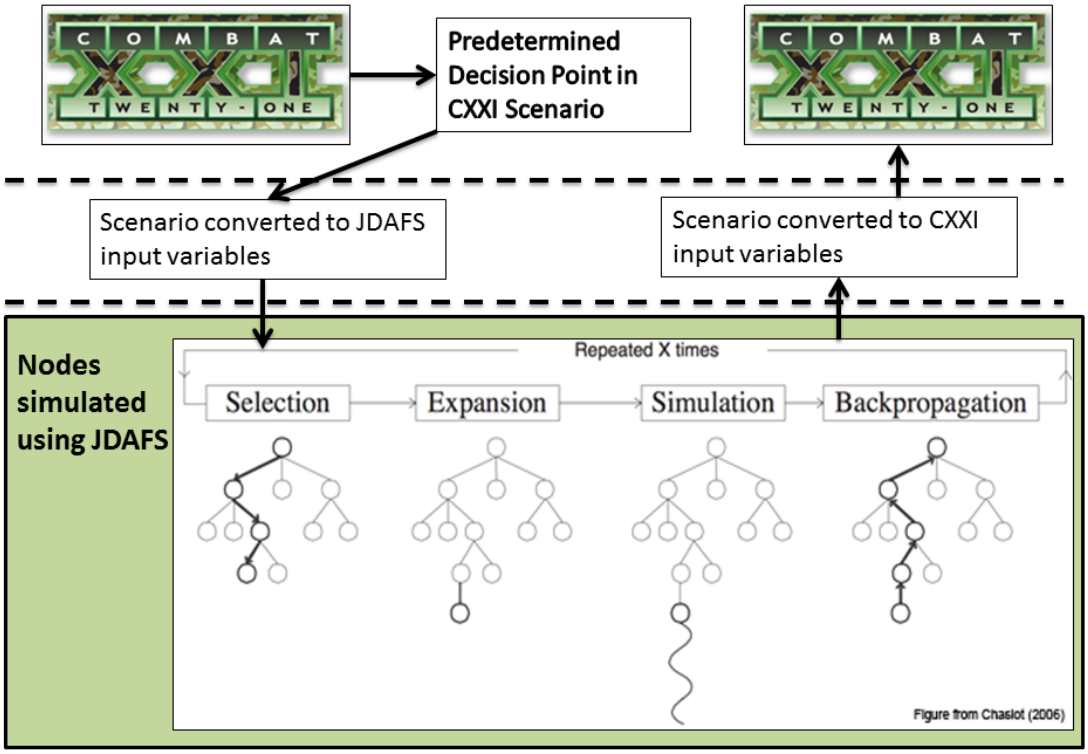

Figure 2. MCTS Implementation Strategy

Implementation Strategy

Once we had our initial prototype we discussed implementation strategy. COMBATXXI is a high resolution simulation in which a small scenario can be computationally expensive. With this consideration in mind we re-examined our fundamental objective of evaluating mission command within the scenario. In order to run the scenario multiple times to build the state space we must consider the computational cost.

Our approach (see figure 2) creates predetermined criteria for decisions within a COMBAT-XXI scenario. When these conditions are met the scenario stops. The current entity state and scenario conditions are transfered into JDAFS, a low resolution simulation which is not computationally expensive. The MCTS algorithm runs, building the state spaces in the tree using the JDAFS simulation.

Once the stopping criteria within the algorithm is reached and a decision is made, the decision is passed back to COMBATXXI for execution by the entity. COMBATXXI continues the simulation executing the decision until completion or stopping criteria is met again.

Implementation Considerations

To implement this strategy, we identified the entity and scenario variables which were necessary to transfer to JDAFS, considering the limitation of the simulator. We first identified what is important to the scenario and what can be left out. We began framing the answer but quickly went outside the capacity of what the team knew of the current programming codes of the simulations.

Realizing our lack of expertise in programming, it was clear there was a contract requirement for programming support for a programmer familiar with the languages used in JDAFS and COMBATXXI. We worked with the NPS contracting cell to announce the requirement which became a lengthy process[12]

Project Termination

While going through the contracting process, we decided not to pursue a contract effort because even if we did, the cost in terms of computing time would have been prohibitive to the point of non-use. This was identified to the sponsor[13] and the lead stakeholder[14] during an In-Progress-Review. They agreed to terminate the project because of the fore-seen limitations to implementing the algorithm in COMBATXXI due to prohibitive computing time even if completed.

Test and Evaluation

This phase was not completed due to the decision to not go forward with contracted programming support.

Analysis

This project is documented in this technical report.

[12]More than 6 months before the performance work statement was posted.
[13]Mr. Paul Works of the Methods and Research Office (MRO)
[14]Mr. Chad Mullis, Director, Models and Simulation, TRAC-WSMR

Chapter 3
Conclusion

The MCTS algorithm has merits for implementation within the simulation environment as an autonomous decision tool to aid in mission command analysis. COMBATXXI, in its current configuration, is not the right platform for MCTS algorithm implementation as concurred by the sponsor and lead stakeholder. The findings of this project support implementing the algorithm using a low resolution simulation, such as JDAFS, to find the "best" decision but implementing this algorithm in a high resolution simulation such as COMBATXXI is not prudent at this time.

This page intentionally left blank.

Appendix A
Original Proposal

Project Title: Analyzing the Impact of Mission Command in Simulation Environments

Sponsor/Manager: TRAC-MRO (Mr. Paul Works)

1 General Information

Government Lead:

MAJ Christopher Marks, TRAC-Monterey, ATTN: ATRC-RDM, 700 Dyer Road, Monterey, CA 93943, 831-656-3751 (DSN 756-3751), FAX 831-656-3084, cemarks@nps.edu.

Resource Management POC:

MAJ Edward Masotti, TRAC-Monterey, ATTN: ATRC-RDM, 700 Dyer Road, Monterey, CA 93943. 831-656-6271 (DSN 756-6271), FAX 831-656-3084 emmasott@nps.edu.

Technical POC:

LTC Jon Alt, Director, TRAC-Monterey, ATTN: ATRC-RDM, 700 Dyer Road, Monterey, CA 93943. 831-656-3086 (DSN 756-3086), FAX 831-656-3084 jkalt@nps.edu.

Project Objective: To demonstrate analysis of mission command in military simulation environments using Monte Carlo Tree Search and other methods from artificial intelligence.

Background: Mission Command is the exercise of authority and direction by the commander using mission orders to enable disciplined initiative within the commander's intent to empower agile and adaptive leaders in the conduct of unified land operations [1]. Military simulation environments for analysis represent entity decision-making to varying degrees, but typically limit decision-making to a set of first order logic rules. This makes it difficult to conduct analysis to understand the value of a decision in a given situation. TRAC-MTRY is wrapping up a six-month effort that applied Monte Carlo Tree Search (MCTS), an AI method for identifying good paths through a decision space, to address several military decision making situations:

- Mission area assignment and scheduling for aerial platforms.
- Weapons selection decisions in COMBATXXI.
- Subordinate element assignments in the Joint Dynamic Allocation of Fires and Sensors (JDAFS) simulation environment.

The output of this initial effort will be several simple MCTS implementations, along with analysis and documentation of the results. Based on these results, the team recommends

further research to apply MCTS methods into more complex scenarios in order to conduct more relevant analyses.

Technical Approach: This follow-on research effort will begin with detailed problem definition work with the sponsor. Following the problem definition phase, the project team will develop simulation use-cases with analysis goals oriented on mission command relevant to TRAC studies and research. These use-cases will include specific implementations of MCTS or a related AI method and will leverage initial implementation work from the previous year's effort conducted in COMBATXXI. The team will develop and execute a design of experiments for each use-case and will provide documentation of all results to the sponsor and a recommendation on the use of these techniques to enable mission command analysis.

2 Milestones and Deliverables

Schedule:

N	Receipt of funds
N+1	Problem definition; initial IPR to sponsor.
N+3	Use-cases & analysis goals identified, IPR to sponsor.
N+6	Use-case scenario files and MCTS implementations complete, IPR to sponsor.
N+9	Experimentation complete, final IPR to sponsor.
N+10	Documentation complete.

Deliverables:

- Deliverable 1. Use case scenario files and updated MCTS implementations.
- Deliverable 2. Design of experiments with results.
- Deliverable 3. Documentation of all analyses and recommendations for analysis of mission command in simulation environments.

3 Project Funding Information

Total Funds: $135,000.

$90,000	NPS faculty
$45,000	TRAC-WSMR Developer

References

[1] Headquarters, Department of the Army. *Army Doctrine Publication (ADP) 6-0; Mission Command*. Government Printing Office, Washington, D.C., May 2012.

Appendix B
Threat Density Map Modeling for Combat Simulations

Threat Density Map Modeling for Combat Simulations

Francisco R. Baez
Christian J. Darken
Modeling, Virtual Environments, and Simulation (MOVES) Institute
Naval Postgraduate School
700 Dyer Road, Watkins Ext. 265
Monterey, CA 93943
831-656-7582
frbaezto@nps.edu, cjdarken@nps.edu

Keywords:
Threat Modeling, Combat Simulations, Probability Theory

ABSTRACT: *The modeling and simulation community has used probability threat maps and other similar approaches to address search problems and improve decision-making. Probability threat maps describe the probability of a location containing one or more enemy entities at a particular time. Although useful, they only describe the likelihood that the location is occupied without addressing the degree to which it is occupied. Thus, we investigate whether threat density maps that describe the searcher's expectation of seeing a number of target agents at a certain location in a given time interval are a viable method for improving synthetic behaviors in combat simulations. As a proof of principle, this paper introduces a probability model which quantifies the searcher agent's subjective belief about the number of enemy entities in a location, given the initial information described by a prior density function and the information provided by the assumed sensing model. In addition, this paper discusses a framework for initializing the model, as well as the model's key advantages and current limitations.*

1. Introduction

A probability threat map is a knowledge representation of the search environment as a discrete probability distribution, which provides a snapshot in time of unobserved threat locations. More specifically, probability threat maps are models of the perceived threat that describe the probability that any given one of a number of unseen entities that are moving independently is in a location (Darken, McCue, & Guerrero, 2010). They have been applied successfully to drive the synthetic behaviors for target scanning in military training simulations by prioritizing locations that are most likely to contain targets (Darken et al., 2010; Evangelista, Ruck, Balogh, & Darken, 2011).

Probability threat maps are derivatives of probabilistic occupancy maps used by game developers for opponent and target tracking (Isla & Blumberg, 2002; Isla, 2006); in addition, they use methods and techniques originally developed for mobile robotics designed to improve localization, search, navigation, and decision-making behaviors (Elfes, 1989; Thrun 2003). Others analogous approaches have been applied to investigate search problems with incomplete and uncertain information using unmanned aerial sensors and autonomous ground sensors (Bertucelli & How, 2005, 2006; Chung & Burdick, 2008, 2012; Chung, Kress, & Royset, 2009; Kagan & Ben-Gal, 2013).

Existing probability threat maps approaches for military simulations (Darken et al., 2010; Evangelista et al., 2011) provide simulated entities with subjective knowledge of likely enemy locations over a defined area, which is then used to carry out search decisions and search behaviors (e.g., select the next search area, modify movement, change tactical formations, path planning). These methods successfully improved the representation of search based on situational awareness and environmental factors in military simulations. However, there is a stated need and interest for expanding these methods essentially to enhance the representation of search, reasoning, and decision-making behaviors in combat simulations.

We believe that the current implementation of probability threat maps could be augmented with additional subjective knowledge of the threat necessary to model and simulate combat scenarios. Probability threat maps use statistical description of likely enemy locations but lack the ability to infer the number of the enemy from observed data and prior information. Ideally, the searcher should gain whatever information he can during the search process and then assess his subjective belief to infer the likely disposition (i.e. location and number of entities) of the threat.

A threat density map is a knowledge representation of the expected number of the enemy entities located inside each subdivision of the simulated area. More specifically, it quantifies the searcher agent's expectation of finding a number of enemy entities at a particular location in a time interval. The purpose threat density maps is to augment combat simulations

with actionable subjective knowledge that can be exploited by the simulated entities for reasoning and decision-making in response to the threat and environment circumstances.

In contrast to probability threat maps, threat density maps provide the searcher agent with additional data needed in combat simulated scenarios to make better decisions amongst different courses of action consistent with the situation presented by the enemy forces (Pew & Mavor, 1998). For instance, depending on the size of the enemy forces the searcher agent can decide whether to defend, assault, attack, withdraw, avoid combat, or bypass. Such decisions would control other behaviors such as searching techniques, path planning, patrolling strategies, etc. In this context, simulated entities would have additional threat knowledge to reason and act upon.

In this paper, we introduce a threat density map model as a proof of principle. We build on current probability threat maps approaches to model the searcher's subjective belief regarding the threat size as a posterior density map instead of a discrete probability distribution. The main contribution of this paper is the formulation of the proposed threat density map model for combat simulations. This introductory section is followed, in Section 2, with a description of the problem and the model formulation. Section 3 describes the advantages and limitations of the current state of the model. Section 4 provides concluding remarks and discusses the direction of the future research.

2. Problem Description and Formulation

A threat density map, $\boldsymbol{tm}$, is a random variable defined over a finite set of locations, X, which assigns a score to each individual cell $x_i \in X$, $i = 1, \dots, C$, at a certain time step t describing the expected number of enemy entities in each cell. The set of locations, X, represents the area of operations discretized into a two-dimensional grid comprising C total cells, which can either be unoccupied or occupied by one or more enemy entities. The random variable $\boldsymbol{tm} = (tm_1, \dots, tm_C)$ denotes the state of the threat density map, where the random variable tm_i indicates the number of enemy entities in cell x_i. Let $k \in \mathbb{Z}^+$ be the grand total number of enemy entities across all cells in X, namely, $k = \sum_{i=1}^{C} tm_i$.

Our fundamental problem is to infer the unknown value of tm_i, namely the unknown number of enemy entities located in the individual cells, based on a sequence of sensing outcomes and assumptions about the success of those sensing actions. To accomplish this, we first initialize each cell with a prior density function, $p(tm_i)$, based on how the searcher believes the enemy is spatially distributed and the certainty of prior information available. This prior information is then combined with the data from our assumed sensing model, $p(s_i^t | tm_i)$, which is the probability density function of the number of enemy entities sensed in cell x_i at time step t, s_i^t, conditional on tm_i. Finally, for each individual cell we update the prior $p(tm_i)$ to the posterior, $p(tm_i | \delta_i^t)$, with the data from the sensing model, $p(s_i^t | tm_i)$, and infer the expected number of enemy entities through successive Bayesian updates.

It is important to define key assumptions required for our framework. First, the total number of enemy entities in the set of locations is a priori unknown but bounded by k enemy entities. Second, the spatial distribution of the enemy entities across the set of locations can be represented with a prior density function. Third, the number of enemy entities in any given cell is independent of the number of entities in all other cells. Lastly, sensing actions within the same cell are conditionally independent from other sensing actions whether in the same cell or in other cells. Clearly, the assumptions of independence and conditional independence may not be realistic as the knowledge that a cell is occupied or not at a particular time can help figure out the state of it and other cells at the current and future times. However, these assumptions, commonly used in related literature, reduce computational complexities and allow us to decompose the problem for solving threat density maps for the individual cells independently (Thrun, 2003; Merali & Barfoot, 2013).

2.1. Initializing Threat Density Maps

To initialize tm_i at time step $t = 0$, we choose a prior density function, $p(tm_i)$, for every cell to represent the searcher's subjective belief about the enemy's spatial distribution in the location set previous to initiating the search. This prior density function summarizes the probability that the random variable tm_i takes on any given values n, which we can write explicitly as $p(tm_i = n)$.

Defining sensible prior density functions varies by the type of prior information (i.e. specific, vague, insufficient) about the enemy and the unknown parameter tm_i. Information regarding the enemy (e.g., size, composition, known or suspected locations, likely formations and movement) normally exists in military scenarios for combat simulations and should be used to initialize priors for each cell. Exact or credible intelligence data available (e.g., intelligence reports, situation reports, satellite imagery) of the enemy and the environment can be useful to define k and strong priors and perhaps to define other aspects of the world (e.g., likely movement routes, probable employment areas, key terrain, obstacles). On the other hand, with

vague intelligence data we might have to assume a prior based on general considerations (e.g. most probable or most dangerous enemy disposition).

In brief, we have distinct cases of prior information available (i.e. specific, vague, and no prior information available) to consider when specifying $p(tm_i)$. The inclusion of prior information into the prior $p(tm_i)$ is one of the benefits of our approach because it leads to stronger inferences about tm_i. Regardless of the level of certainty, we can specify a prior to quantify uncertainty around the spatial distribution of the enemy entities and express what is believed or known about tm_i before inspecting any cell $x_i \in X$. Below we discuss a discrete prior density function that can be used to initialize the $p(tm_i)$ with prior information.

2.1.1. Discrete Prior Density Function

Consider the case in which the search agent lacks or has vague prior information. A common practice in such situation is to define a conventional prior, such as the discrete uniform, that does not favor any particular value. However, as previously mentioned prior information for combat simulation scenarios is typically available. Therefore, it is then sensible to define a prior density function that can account for a broad range of possibilities fundamental to combat simulated scenarios.

For this particular problem, with no idea about the distribution of tm_i we define a discrete prior assuming that any cell in X could contain up to k targets evenly distributed but more likely for the enemy to be nonexistent in some cells. Accordingly, tm_i is a discrete random variable with a finite range bounded by k, i.e., $\{0, 1, \dots, k\}$. Further, we assume that the prior is defined for each cell such that $p(tm_i) = 1/k$ for $n = 1, \dots, k$. However, our prior subjective belief inclines us to anticipate that many cells will be empty rather than occupied because enemy forces tend to cluster together whether they operate as cohesive large element or as smaller dispersed elements. To represent such belief we define the parameter ε such that each of the values in the range $1 \le n \le k$ occurs with probability $\varepsilon(1/k)$ and $(1-\varepsilon)$ for $n = 0$. That is, the unconditional prior probability distribution for an individual cell is given by the following probability density function:

$$p(tm_i = n) = \begin{cases} \varepsilon\left(1/k\right), & n = 1,\ 2,\ \dots,\ k \\ 1-\varepsilon, & n = 0 \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

The expected value of the random variable tm_i for cell x_i at time step $t = 0$ is:

$$\mu_{tm_i} = E(tm_i) = \sum_{n=1}^{k} n\varepsilon\left(1/k\right) = \varepsilon\left(k + 1/2\right) \quad (2)$$

Although the choice of ε is subjective it is also suitable to initialize $p(tm_i)$ when specific prior information is available. For instance, suppose we know the mean number of enemy entities for some specific cells. In this case, we do not have any difficulty incorporating this information in $p(tm_i)$. We simply solve Eq. (2) for ε, i.e., $\varepsilon = 2\mu/(k+1)$, for $\varepsilon \in [0,1]$, and use this value to define the prior of tm_i for those particular cells.

2.2. Sensing Model

Sensing actions, namely, observing or inspecting cells, are knowledge-producing events that changes the searcher's subjective belief of the threat. The searcher's ability to observe enemy entities in a cell is modeled using the combat simulation's target detection model, which specifies the probability of detecting a target, P_d, as a function of the brightness of the target, the brightness of the target's background, and the subjective size of the target given that one or more targets are present in the location. Although P_d varies by type of target, it is generally constant for targets of the same type and size, and against a particular background.

In our framework, sensing actions represent binomial trials with $(k+1)$ possible outcomes (i.e. observing between zero and k enemy entities) of the actual number of entities in the cell. They return the number of enemy entities sensed, s_i^t, in cell x_i at time step t. Therefore, we specify a binomial sampling model, $p(s_i^t | tm_i)$, which describes the searcher's ability to gain subjective knowledge regarding tm_i. This sampling model provides the conditional probability that s_i^t is b conditioned on tm_i and given P_d, i.e., $p(s_i^t | tm_i) = p(s_i^t = b | tm_i = n)$, expressed as

$$p\left(s_i^t = b \middle| tm_i = n\right) = \frac{n!}{b!(n-b)!}\left(P_d\right)^b\left(1-P_d\right)^{n-b} \quad (3)$$

for $b = 0, 1, \dots, k$ and $0 \le P_d \le 1$. In Eq. (3) the binomial coefficient $n!/b!\,(n-b)!$ describes the number of combinations of n things taken b at a time without regard of their order; $(P_d)^b$ is the likelihood of b detections given P_d; and $(1-P_d)^{n-b}$ is the probability of missing $(n-b)$ of the possible detections.

2.3. Multiple Sensing Actions

Above we focused on the probability for a single sensing action at time step t. However, our goal is to infer tm_i based on all cell inspections through time step t. Let $p(\delta_i^t|tm_i)$ indicate the distribution of the sensing outcomes for cell x_i up to time step t and let $\delta_i^t = \{s_i^{\tau_1}, \dots, s_i^{\tau_j}\}$ denote the history of the number of enemy entities sensed through time step t. Assuming multiple inspections of cell x_i at different time steps, $s_i^{\tau_j}$, $j = 1, \dots, t$, represents the number of enemy entities sensed at time τ_j.

As previously stated, the probability of each sensing action is conditionally independent of other sensing actions; specifically, s_i^t and δ_i^{t-1} are conditionally independent given tm_i. In other words, if tm_i is known, additional knowledge of δ_i^{t-1} does not change the searcher's belief about how many enemy entities he will see at the next observation (s_i^t). Therefore, the probability of the data set (i.e. history of the enemy entities sensed) is given by:

$$p\left(\delta_i^t \middle| tm_i\right) = p\left(s_i^t, \delta_i^{t-1} \middle| tm_i\right) = p\left(s_i^t \middle| tm_i\right) p\left(\delta_i^{t-1} \middle| tm_i\right) \quad (4)$$

2.4. Updating Threat Density Maps

We now discuss how to update probabilities after a new sensing action is performed. According to Bayesian inference, we can estimate the posterior through time step t, $p(tm_i|\delta_i^t)$, given a prior on tm_i and the data resulting from the sensing model, $p(s_i^t|tm_i)$. Applying Bayes rule to the terms $p(\delta_i^t|tm_i)$ and $p(\delta_i^{t-1}|tm_i)$ in Eq. (4) and with the conditional independence assumption of the sensing actions results in the posterior density function $p(tm_i|\delta_i^t)$ of tm_i given the history of enemy entities sensed in cell x_i through time step t. The posterior density is given by

$$p\left(tm_i \middle| \delta_i^t\right) = \frac{p\left(s_i^t \middle| tm_i\right) p\left(tm_i \middle| \delta_i^{t-1}\right)}{p\left(s_i^t \middle| \delta_i^{t-1}\right)} \quad (5)$$

where $p(s_i^t|tm_i)$ is obtained from the sensing model in Eq. (3); $p(tm_i|\delta_i^{t-1})$ represents either a prior at time step $t = 0$, i.e., $p(tm_i)$, or a posterior without the most recent sensing action result; and $p(s_i^t|\delta_i^{t-1})$ is the normalization factor resulting from marginalizing over tm_i and applying the foregoing conditional independence assumption of sensing actions given tm_i:

$$p\left(s_i^t \middle| \delta_i^{t-1}\right) = \sum_{n=0}^{k} p\left(s_i^t \middle| tm_i = n\right) p\left(tm_i = n \middle| \delta_i^{t-1}\right). \quad (6)$$

Substituting Eq. (6) in Eq. (5), the individual cell beliefs can be updated using the following:

$$p\left(tm_i \middle| \delta_i^t\right) = \frac{p\left(s_i^t \middle| tm_i\right) p\left(tm_i \middle| \delta_i^{t-1}\right)}{\sum_{n=0}^{k} p\left(s_i^t \middle| tm_i = n\right) p\left(tm_i = n \middle| \delta_i^{t-1}\right)}. \quad (7)$$

Eq. (7) results in a distribution of the unknown number of enemy forces in the cell conditioned on the observed sample data. Thus we have a probability model that quantifies the searcher's new state of subjective belief about tm_i, given the initial information described by the prior $p(tm_i)$ and the information provided by the sensing model $p(s_i^t|tm_i)$.

2.5. Inference about the Number of Enemy Entities

During initialization we estimate the expected number of enemy entities for every cell from the prior probabilities and maintain this during runtime until the cell posterior distribution is updated after a sensing action. Once the posterior is computed, we utilize Eq. (8) to determine the expected number of entities x_i.

$$E\left(tm_i \middle| \delta_i^t\right) = \sum_{n=0}^{k} n p\left(tm_i = n \middle| \delta_i^t\right). \quad (8)$$

3. Advantages of Threat Density Maps

In this section we discuss the advantages of providing simulated entities with threat density maps as well as the limitations of the current state of the model. Simply put, the main advantage of the proposed approach as compared to probability threat maps is that a threat density map provides a probability distribution of the unknown number of enemy entities and the expected number of enemy entities in a cell, which can be influenced by a detailed prior distribution. To conceptualize the notion of threat density maps applied to combat simulations and to demonstrate its practicality and advantages, we coded and implemented in a rudimentary JavaScript simulation the aforementioned threat density map and for comparison, an adaptation of the probability threat maps (see Appendix 1) discussed in Darken et al. (2010).

The notional scenario consists of a simulated infantry soldier (searcher) searching for an enemy fireteam to either engage them or to report their disposition, location, and actions. From intelligence data the searcher knows that enemy fireteam (targets) is not moving and consists of three entities close together and one scout far ahead. Figure 2(a) shows the targets actual distribution, i.e., $\{x_{11} = 1, x_{16} = 3\}$, which is

unknown to the searcher. Based on their doctrinal spatial dispersion and the size of a cell we initialized threat density maps for the individual cells assuming that any cell could contain one or three targets but not two or four, yet being free of enemy entities is even more probable than occupation by one or three. Figure 1 shows an example of this prior for a single cell. Finally, we assumed a uniform prior to initialize the probability threat map and the probability of detection remained constant for the simulation, i.e., $P_d = 0.65$.

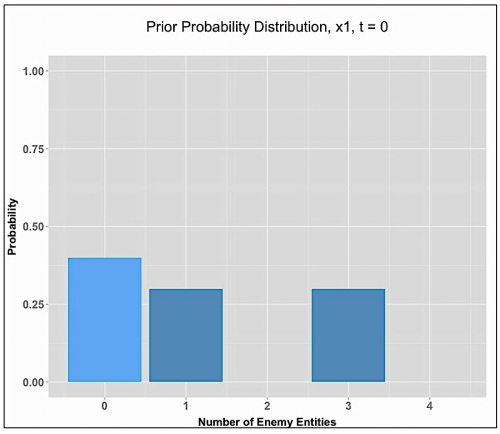

Figure 1: Discrete prior distribution of tm_1 for cell x_1 with $\varepsilon = 0.4$ assuming that it is more likely that the cell is occupied (containing either one or three targets) than empty.

One of the main advantages of our Bayesian approach to threat density maps is the availability of a posterior distribution of the unknown number of targets in a cell rather than a single value as in the probability threat map approach. For example, consider the situation shown in Figure 2 in which the searcher sensed zero targets after inspecting cell x_1. The low probability value in the probability threat map [Figure 2(b)] indicates that cell x_1 is less likely to contain one or more targets when compared to the other cells. However, the searcher lacks knowledge about the degree to which the cell x_1 is occupied, when in fact it can be empty or occupied by one or three targets because cell inspections are not perfect. The coarse threat knowledge provided by the probability threat map, although useful for search decisions is not sufficient for making decisions related to tactical courses of action.

On the other hand, the threat density map posterior distribution summarizes the state of knowledge about the unknown number of targets in the cell conditional on the prior and sensing data. In contrast to the probability threat map, the threat density map in Figure 2(d) suggests that although cell x_1 is more likely to be empty there is still a chance to find one or three targets in the cell. In this situation, the posterior distribution of tm_1 provides the searcher with a more accurate picture of the likely state of cell x_1. This more detailed representation of threat knowledge provides the searcher the basis for a more confident course of action selection.

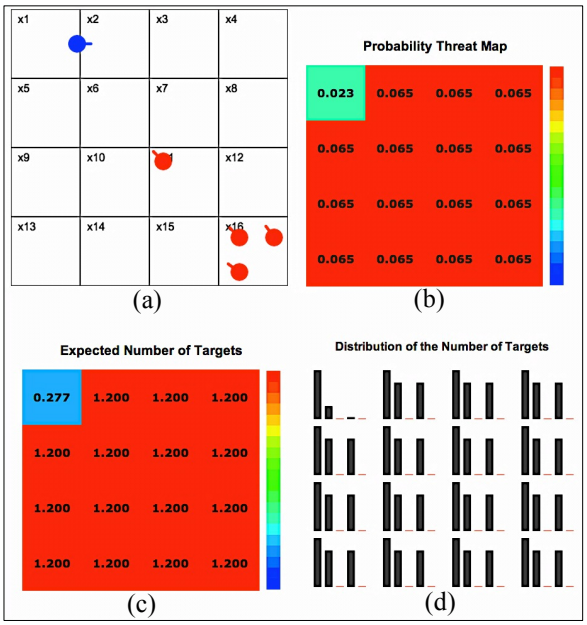

Figure 2: Screenshot of the simulated scenario at time step $t = 0.25$ where the searcher is depicted in blue and the targets are depicted in red (a), the probability threat map (b) and threat density map consisting of the expected number of targets (c) and the related probability distributions of the number of targets (d).

Consider the situation in Figure 3 in which the searcher after inspecting several cells sensed two targets (dark red entities) in cell x_{16}. For such situation, it would be difficult for the searcher to select a course of action that provides the best possibility of success based solely on the probability threat map. Therefore, it is appealing to quantify the searcher's expectation of finding a number of targets at the cell. Updating the threat density map's prior information with sensed data, provides interpretable answers, such as the event that tm_{16} equals three has probability of one [Figure 3(d)] thus, the searcher could expect to see three targets in the cell [Figure 3(c)]. Then, he can exploit this subjective knowledge to make reasonable decisions consistent with the likely state of the threat, for example, decide to search the cell for the unobserved target or to move out of the cell and avoid combat.

Likewise, threat density map data can also be used to support reasoning. Consider a separate simulation run (Figure 4) in which the searcher sensed one target (dark red entity) in cell x_{11} given $P_d = 0.9$. Based on the threat density map the searcher could assume with a high degree of certainty that he found the scout entity of the enemy fireteam and hence could use this belief for identifying the neighboring cell that could contain

the remaining three targets and to determine how he deploys, orient, and engages the remaining targets.

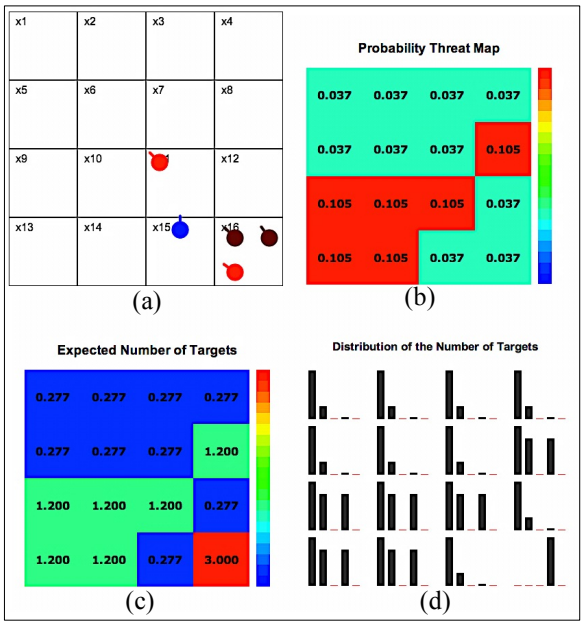

Figure 3: Screenshot of the scenario and the state of subjective threat knowledge in which the searcher sensed two targets (depicted in dark red) in cell x_{16}.

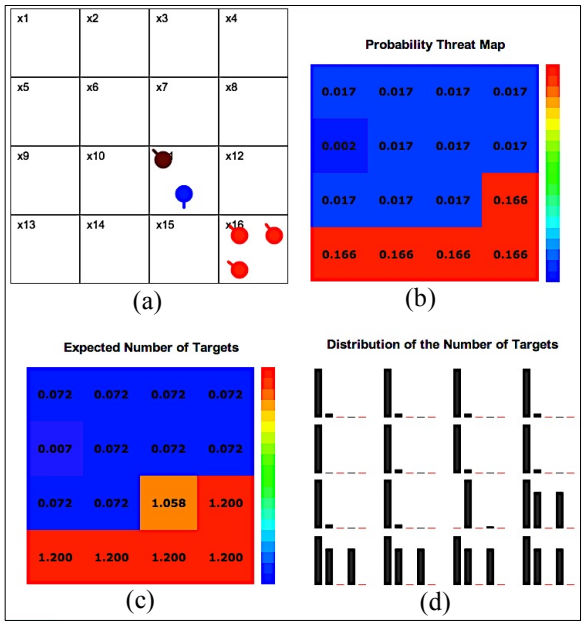

Figure 4: Screenshot of the scenario and the state of subjective knowledge in which the searcher sensed one target (depicted in dark red) in cell x_{11}.

3.1. Integrating Prior Information

The incorporation of a prior density function for tm_i with prior information is the final favorable feature of the threat density map that differentiates it from the probability threat map. As previously mentioned, intelligence data or prior information is typically available for combat simulated scenarios. Regardless of the level of certainty of the prior information, we can use the aforementioned discrete prior density function or other suitable discrete distributions to describe uncertainty for tm_i in a mathematical model. However, from a modeling perspective the difficulty is in how to effectively integrate prior information from different sources (e.g., intelligence, doctrine, environment) using a prior density function (Blasco, 2007).

In Figure 1 above we already demonstrated an example for initializing threat density maps given prior information and intelligence data (i.e. the total number of targets and their tactical formation). Below we briefly discuss two cases of prior information available common to combat simulated scenarios for initializing threat density maps.

First, presume that the prior information available consists only of the total number of enemy entities (a fireteam of four entities) and their posture (not moving) but neither their actual location nor their tactical formation is known. In this situation of vague prior information is sensible to assume that any cell could contain up to four enemy entities and logically we can expect that many cells will be empty instead of occupied. Accordingly, we could set the value of ε to be 0.75 and utilize Eq. (1) to initialize threat density maps for each cell $x_i \in X$. Figure 5 shows the prior distribution for cell x_1.

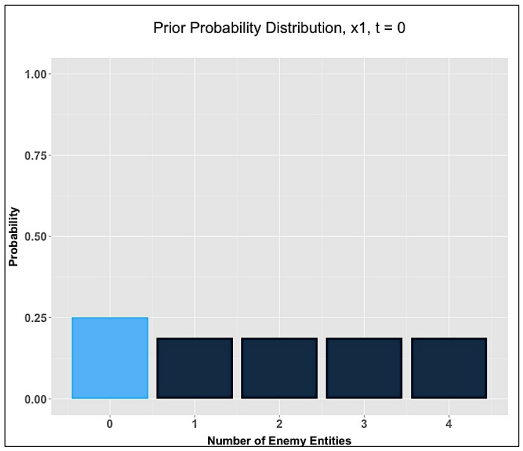

Figure 5: Discrete prior distribution of tm_1 for cell x_1 assuming that it is more probable to be empty and equally likely to be occupied by at least one and no more than four targets.

The plot in Figure 5 shows that it is more likely for a cell to be unoccupied and equally possible to be occupied by one, two, three, or four enemy entities. The expected number of enemy entities in each cell is 1.875, thus the searcher can expect to find approximately two enemy entities in any particular cell at the next time step.

Second, specific prior information can easily be incorporated through the prior density function. For example, suppose that from the most current intelligence data available it is known that there is a squad-size element in a linear defense, arrayed from the southwest corner of the area of operations to the northeast corner, heavily concentrated in cell x_{11}, defending the southeast sector of the area of operations, as depicted in Figure 6. Incorporating this prior information into the model can be done in a flexible manner and inferences can be compared under different priors in order to choose a prior that characterizes the most likely threat situation. One alternative, for example, is to set the value of ε equal to one for the cells known to be occupied and zero otherwise. Such an approach can be efficient but it does not account for the possibility that the situation could change before the searcher reaches any of these cells. Therefore, one could select other values of ε for the cells of interest. Perhaps another alternative is to deduce from doctrine and terrain data the maximum number of enemy entities that can occupy a cell and other relevant factors to initialize the priors for each cells that produce $E(tm_{8,14}) = 2$, $E(tm_{11} = 5)$, and zero otherwise.

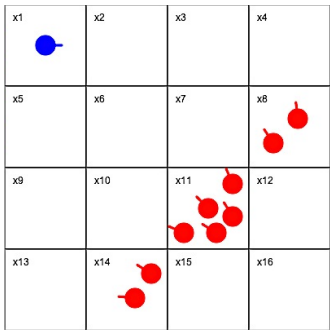

Figure 6: Screenshot of the location set with ground truth data. The searcher is depicted in blue and the targets in a linear defense, heavily concentrated in cell x_{11}, are depicted in red.

3.2. Current Limitations of Threat Density Maps

As we have seen in the previous examples, there are significant advantages of augmenting combat simulated scenarios with threat density maps as they provide simulated entities with actionable subjective knowledge to make course of action decisions, which in turn determines other search, movement, and path planning behaviors. However, the proposed approach has some fundamental limitations. While the assumptions of independence and conditional independence, described in Section 2, allows us to solve the threat density maps for the individual cells, the model excludes features for modeling spatial dependencies and temporal effects. This limitation is evident in Figure 3(c) and 3(d) as the model properly estimates the expected number of enemy entities in the cell, i.e. $E(tm_{16}) = 3.0$, essentially due to the inclusion of prior information into the model; however, it fails to exploit this information for estimating tm_i for the other cells.

4. Conclusions and Future Directions

In this paper we proposed a threat modeling approach for estimating the number of the enemy entities at a certain location in a given time interval. The model estimates the expected number of enemy entities as a posterior density map, can be initialized with intelligence reports and prior information, and works for any number of enemy entities and their spatial distribution. Although a threat density map approach is not required for all combat simulation models and scenarios, they offer several important advantages over probability threat maps that make them suitable for implementation in combat simulations for improving the representation of search, reasoning, and decision-making behaviors.

Efforts are underway to introduce probability distributions that can model threat movement. Furthermore, future work will focus on addressing known limitations and extending the proposed model by introducing spatial and temporal dependencies and interactions, and developing hierarchical threat density map representations. Finally, we plan to experiment with and characterize the utility of the model for improving the capabilities of simulated entities in a combat simulation scenario for different threat conditions.

6. References

Bertuccelli, L. F., & How, J. P. (2005). Robust UAV search for environments with imprecise probability maps. *44th IEEE Conference on Decision and Control, and the 2005 European Control Conference,* 5680-5685.

Bertuccelli, L. F., & How, J. P. (2006). Search for dynamic targets with uncertain probability maps. *Proceedings of the 2006 American Control Conference*.

Blasco, A. (2007). Bayesian statistic course. Retrieved from the web October 15, 2013 http://mastergr.webs.upv.es/Asignaturas/Apuntes/08. Cuantitativa 3/LECTURE NOTES.pdf

Chung, T. H., & Burdick, J. W. (2008). Multi-agent probabilistic search in a sequential decision-theoretic framework. *IEEE International Conference on Robotics and Automation*, 146-151.

Chung, T. H., & Burdick, J. W. (2012). Analysis of search decision making using probabilistic search strategies. *IEEE Transactions on Robotics*, *28*(1), 132-144.

Chung, T. H., Kress, M., & Royset, J. O. (2009). Probabilistic search optimization and mission assignment for heterogeneous autonomous agents. *IEEE International Conference on Robotics and Automation*, 939-945.

Darken, C. J., & Anderegg, B. G. (2008). Particle filters and simulacra for more realistic opponent tracking. *Game AI Programming Wisdom 4*.

Darken, C. J., McCue, D., & Guerrero, M. (2010). *Realistic fireteam movement in urban environments*.

Elfes, A. (1989). Using occupancy grids for mobile robot perception and navigation. *Computer*, *22*(6), 46-57

Evangelista, P. F., Ruck, J., Balogh, I., & Darken, C. J. (2011). Visual awareness in combat models. *Proceedings of the 20th Behavior Representation in Modeling and Simulation*.

Isla, D. A., & Blumberg, B. M. (2002). Object persistence for synthetic creatures. *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems*, 1356-1363.

Isla, D. (2006). Probabilistic target tracking and search using occupancy maps. *AI Game Programming Wisdom*, *3*, 379-388.

Kagan, E., & Ben-Gal, I. (2013). Problem of search for static and moving targets. In *Probabilistic Search for Tracking Targets: Theory and Modern Applications*. Chichester, UK: John Wiley & Sons, Ltd.

Merali, R.S., & Barfoot, T. D. (2012). Patch map: A benchmark for occupancy grid algorithm evaluation. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),* 3481-3488.

Pew, R. W., & Mavor, A. S. (1998). *Modeling human and organizational behavior: Application to military simulations.* National Academies Press.

Thrun, S. (2003). Learning occupancy grid maps with forward sensor models. *Autonomous robots*, *15*(2), 111-127.

Appendix 1: Probability Threat Map Adaptation

In this section we briefly describe our basic adaptation of the probability threat maps approach discussed in Darken et al. (2010).

Let q_i be the conditional probability that an unseen enemy entity is present in cell $x_i \in X$ and after inspecting cell x_j, where $x_i \neq x_j$, $\tilde{q}_i$ is the estimated probability before inspecting cell x_i, and P_d is the probability of detecting a target (see Section 2.2). According to the axioms of probability theory, $0 \leq q_i \leq 1$ and the total probability over all C cells is $\sum_{i=1}^{C} q_i = 1$. Suppose the searcher inspects cell x_j, assuming that cell inspections are independent of neighboring cells, then, q_i takes the form

$$q_i = \frac{\tilde{q}_i I_i + \tilde{q}_j (1 - P_d)(1 - I_i)}{\sum_{i'=1}^{C} \tilde{q}_{i'} I_{i'} + \tilde{q}_j (1 - P_d)} \tag{9}$$

where the term I_i is an indicator function that equals to zero if $x_i = x_j$ and equals to one otherwise.

Author Biographies

Francisco R. Baez is a Lieutenant Colonel in the U.S. Army where he serves as an Operations Research (OR) Analyst. He is currently a Ph.D. student at the U.S. Naval Postgraduate School's (NPS) MOVES Institute, holds an M.S. in OR from NPS, and a B.S. in Logistics from the University of Puerto Rico. His email is francisco.r.baez.mil@mail.mil.

Christian J. Darken, Ph.D., is an Associate Professor of Computer Science at NPS, where he is an affiliate of the MOVES Institute. His current research interests include human behavior simulation for military applications and approaches to artificial intelligence inspired by developmental learning. His email is cjdarken@nps.edu.

This page intentionally left blank.

Appendix C
Mission Command Analysis Using Monte Carlo Tree Search in JDAFS

Introduction

The TRAC Methods and Research Office has initiated an effort to improve analysis methodology for military operations employing network enabled mission command. Representation of network enabled operations has improved significantly in military simulation models over the past decade and a half; however, several key challenges remain. Foremost, is the need to rapidly produce relevant analysis accounting for the operational effects of network-enabled capabilities supporting mission command.

TRAC-Monterey is carrying out supporting research to produce a documented and tested methodology that applies Monte Carlo Tree Search methods to decision situations in order to expand mission command oriented analysis. Mission command features decentralized execution with subordinate commanders exercising disciplined initiative while acting aggressively and independently to accomplish the mission within the commander's intent. The methodology will improve analysis by extending data developed from operational data, wargames, and other subject-matter expertise elicitation into a simulation environment where more extensive and rigorous analysis can be accomplished.

Scope of Work

The scope this work was to design an implementation of MCTS method in the Fires Allocation (FA) scenario in the Joint Dynamic Allocation of Fires and Sensors (JDAFS) simulation environment. Programming and testing of the implementation was left to future work.

The next section will give a brief overview of the JDAFS simulation, and following that, a brief overview of the Monte Carlo Tree Search algorithm. Following that will be a description of the design for adding MCTS to JDAFS for fires allocation. Finally, next steps will be discussed.

Joint Dynamic Allocation of Fires and Sensors (JDAFS)

The Joint Dynamic Allocation of Fires and Sensors (JDAFS) is a low-resolution, constructive entity-level simulation framework that can be rapidly configured and executed and allows force-on-force analysis of differing UAS mixes. This model can simulate UAS operations with optimization in the loop to gain greater insight into UAS interactions and interactions between UASs and sensor targets for a given NAI.

JDAFS provides the ability to conduct quick operational analyses of Joint and Army assets by proving a model that is extremely flexible, configurable, and enables an analyst to very quickly create a simulation model that captures the first-order effects of a scenario. Currently, JDAFS represents aircraft schedules, but does not adequately represent deconfliction of air assets. JDAFS is also ideally well suited for establishing Joint Starting Conditions for any given scenario. JDAFS provides an effective simulation modeling tool to provide quick-turn analysis of capability to compare operational policies and control measures in order to identify those polices and measures which provide the greatest operational performance, focusing primarily on fires and effects.

Fires allocation

A shooter platform has its fires directed by an instance of a Constrained Value Optimizer (CVO). The CVO is so-named because its initial implementation is to assign shooters to targets based on solving an optimization problem. The default CVO in JDAFS formulates an integer linear program based on which targets have been detected and which shooters are available. Since a shooter may have several types of munitions, the assignments are based on the properties of the munition-target pairings. For each pair, a coefficient is calculated for the objective coefficient in the linear program. The algorithm for this calculation is performed in an instance of a Value of Potential Assignment (VPA) class. The default one is described here, but the software has been written so that different computations could be made.

The default VPA calculates the expected net value of engaging a shooter platform with a given weapon i against a given target j: $c_{ij} = v_j p_{ij} - v_i p_{ji}$ where v_j is the value of the shooter platform of weapon i, p_{ij} is the probability that target j is killed by munition i, v_i is the value of the shooter platform of weapon i and p_{ji} is the probability that target j kills

the shooter of munition *i* when they engage. Thus, the optimization problem to be solved is:

$$\max \sum_{i,j} c_{ij} x_{ij}$$

$$\text{subject to}$$

$$\sum_{j} x_{ij} \le A_i$$

$$\sum_{i} x_{ij} \le B_j$$

$$\sum_{i} x_{ij} \ge C_j$$

$$x_{ij} \in \{0,1\}$$

where A_i is the maximum number of targets that can be assigned to shooter i, B_j is the maximum number of shooters that can be assigned to target j, and C_j is the minimum number of shooters that must be assigned to target j. Note that the formulation is totally unimodular, and therefore the linear programming relaxation gives the optimal solution.

Monte Carlo Tree Search (MCTS)

Monte Carlo Tree Search (MCTS) is a method for finding optimal or near-optimal solutions by using Monte Carlo random sampling of potential decisions and building a search tree. The tree is used to evaluate the value of decisions, informed by the results of exploring various branches of the tree. When a pre-determined computational budget has been reached, the algorithm terminates with the action with the highest value is selected as the next "move." Each node of the tree contains the (current) value of the node and the number of times it has been visited. The general form of the algorithm proceeds in four steps (Browne, *et al*, 2012):

1. *Selection*. A child node that is expandable (i.e. is a non-terminal state and has unvisited children) is selected.
2. *Expansion*. One (or more) child nodes are added to the selected node based on feasible actions.
3. *Simulation*. A simulation is executed from each new node according to the default policy (described below) to produce an outcome.

4. *Backpropagation*. Each parent node of the selected one is updated to reflect the outcome.

Two policies are applied that customize the general algorithm; these are:

1. The *tree policy* determines which node is selected during the Selection step.
2. The *default policy* determines how the game is played out from a given node to produce a value.

The general MCTS approach can therefore be summarized by the following pseudo-code (Browne, et al, 2012):

create root node v_0 with state s_0
while within computational budget **do**
$v_l \leftarrow TreePolicy(v_0)$
$\Delta \leftarrow DefaultPolicy(s(v_l))$
$Backup(v_l, \Delta)$
return $a(BestChild(v_0)$

The Tree Policy that will be used for JDAFS is based on the *Upper Confidence Bounds for Trees* (UCT) algorithm (Browne, et al, 2012). This selects the best child v_j of node *v* based on the following formula:

$$UCT_j = \frac{Q(v_j)}{N(v_j)} + 2C_p\sqrt{\frac{2\ln N(v)}{N(v_j)}},$$

where *N(v)* is the number of times node *v* has been visited, v_j is a child node of *v*, and *Q(v)* is the current value of node *v*. The node v_j with the largest value of UCT_j is the one selected next.

Application of Monte Carlo Tree Search to JDAFS

The logical place in JDAFS to apply the MCTS algorithm is for the allocation of fires. That is, to develop a replacement Constrained Value Optimizer (CVO) for JDAFS based on the MCTS algorithm described above.

The implementation of a MCTS CVO would necessarily take into account the subsequent possible behavior of the model following the allocation. Specifically, the "simulation" stage of MCTS would involve replication s of simulating the subsequent

battle given a particular allocation. This has the potential of superior allocations, since the current default CVO is based on a static model.

Implementing the simulation stage of MCTS involves having to solve several coding issues. First, JDAFS would need a mechanism to save the current "real" state of the model so it could be returned to following the output of the MCTS algorithm. It would need to be saved as an initial state for the simulation replications as well. Currently JDAFS does not support for this capability.

Second, a means to restore and recover the state of the individual entities would need to be implemented, since the MCTS simulation would necessarily be modifying those states.

Third, there are units in JDAFS that are not necessarily detected, and so would not be in the initial allocation list, yet would impact the simulation going forward. How to incorporate those undetected entities is a problem that would have to be solved.

Finally, to fully take into account uncertainties about the enemy force, there would have to be a mechanism for forecasting what (undetected) enemy forces might exist and incorporating such pseudo-entities into the simulation stage of MCTS.

References

[1] Buss. A. and P. Sanchez. "Simple Movement and Detection in Discrete Event Simulation," *Proceedings of the 2005 Winter Simulation Conference*, M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, eds.

[2] Buss, A.H. and D. Ahner. "Dynamic Allocation Of Fires And Sensors (DAFS): A Low-Resolution Simulation For Rapid Modeling," *Proceedings of the 2006 Winter Simulation Conference* L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, eds. , Monterey, CA.

[3] Browne, C., *et al* "A Survey of Monte Carlo Tree Search Methods," IEEE Transactions on Computational Intelligence and AI in Games, Vol. 4, No. 1, March 2012.

[4] Phillips, D. and J. Jackson. "Using a Low Resolution Entity Level Modeling Approach," *Phalanx* Volume 38 Number 2, June 2005, p. 15.

Appendix D
References

[1] MAJ Christopher Marks et al. *Mission Command Analysis Using Monte Carlo Tree Search*. Tech. rep. TRAC-M-TR-13-050. 700 Dyer Road Monterey, California 93943: TRADOC Analysis Center - Monterey, 2013. URL: "https://ako.hq.tradoc.army.mil/sites/trac/MTRY/SitePages/Home.aspx".

[2] CJ Ancker. "A proposed foundation for a theory of combat". In: *Naval Research Logistics (NRL)* (1995). URL: http://onlinelibrary.wiley.com/doi/10.1002/nav.3220420303/full.

[3] CB Barfoot. "Markov duels". In: *Operations Research* (1974). URL: http://or.journal.informs.org/content/22/2/318.short.

[4] RH Brown. "Theory of combat: The probability of winning". In: *Operations Research* (1963). URL: http://or.journal.informs.org/content/11/3/418.short.

[5] Y. Friedman. "Optimal strategy for the one-against-many battle." In: *Operations Research* ().

[6] J. Gittins, K. D. Glazebrook and R. Weber. *Multi-armed Bandit Allocation Indices*. 2nd. United Kingdom: Wiley, 2011. URL: http://amstat.tandfonline.com/doi/abs/10.1080/00401706.1991.10484891.

[7] K. Kikuta. "A note on the one against many battle." In: *Operations Research* 31.5 (1983), pp. 952–956.

[8] M Kress. "Modeling armed con icts." In: *Science*, 336.6083 (2012), pp. 865–869.

[9] M Kress. "A many-on-many stochastic duel model for a mountain battle". In: *Naval Research Logistics* (1992). URL: http://faculty.nps.edu/mkress/docs/AMany-On-ManyStochasticDuelModelforaMountainBattle.pdf.

[10] M Kress and I Talmor. "A new look at the 3: 1 rule of combat through Markov stochastic Lanchester models". In: *Journal of the Operational Research ...* (1999). URL: http://www.ingentaconnect.com/content/pal/01605682/1999/00000050/00000007/2600758.

[11] FW Lanchester. *Aircraft in warfare: The dawn of the fourth arm*. London: Constable, 1916. URL: http://books.google.com/books?hl=en\&lr=\&id=fIZCAAAAIAAJ\&oi=fnd\&pg=PR21\&dq=Aircraft+in+Warfare:+The+Dawn+of+the+Fourth+Arm.\&ots=Mfbep6A1Df\&sig=DMBLkAd9DocvqSLT3otxWqtKWbc.

[12] S. M. Ross. *Introduction to stochastic programming*. New York: Academic Press, 1983. URL: http://books.google.com/books?hl=en\&lr=\&id=cfrMw9crazsC\&oi=fnd\&pg=PR7\&dq=Introduction+to+Stochastic+Dynamic+Programming.\&ots=jTejUUrKZZ\&sig=TtK8DLfIcywjgeJmdehRbYrdl-M.

[13] AR Washburn. *Two-person zero-sum games*. 3rd. Rockville: INFORMS, 2003. URL: http://books.google.com/books?hl=en\&lr=\&id=rSW14PzV9qwC\&oi=fnd\&pg=PA1\&dq=Two-Person+Zero-Sum+Games.\&ots=LvaPWpAetL\&sig=6vggAblZLd_N9eBSqZIWEN0p-4o.

[14] A Washburn and M Kress. *Combat modeling*. Springer, 2009. URL: http://books.google.com/books?hl=en\&lr=\&id=G97TQWrykd4C\&oi=fnd\&pg=PA1\&dq=Combat+Modeling\&ots=_jlDszrST5\&sig=Srh7lgCIuc7GoCHSGgffxVTK3xI.

[15] T Williams and CJ Ancker. "Stochastic duels". In: *Operations Research* (1963). URL: http://or.journal.informs.org/content/11/5/803.short.

[16] Mark HM Winands, Yngvi Björnsson, and Jahn-Takeshi Saito. "Monte-carlo tree search solver". In: *Computers and Games*. Springer, 2008, pp. 25–36.

Appendix E
Glossary

COMBATXXI	Combined Arms Analysis Tool for the 21st Century
EEA	Essential Elements of Analysis
JDAFS	Joint Dynamic Allocation of Fires and Sensors
MCTS	Monte Carlo Tree Search
MOVES	Modeling, Virtual Environments, and Simulation
NPS	Naval Postgraduate School
TRAC	Training and Doctrine Command Analysis Center
TRAC-MRO	Training and Doctrine Command Analysis Center Methods and Research Office
TRAC-MTRY	Training and Doctrine Command Analysis Center---Monterey
TRAC-WSMR	Training and Doctrine Command Analysis Center---White Sands Missile Range
TRADOC	Training and Doctrine Command