AFRL-RI-RS-TR-2014-013

PROBABILISTIC DEVIATION DETECTION AND OPTIMAL THRESHOLDS

BAE SYSTEMS, INC.

JANUARY 2014

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the 88th ABW, Wright-Patterson AFB Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2014-013 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/ S /
KURT LACHEVET
Work Unit Manager

/ S /
JULIE BRICHACEK, Chief
Information Systems Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
JANUARY 2014	FINAL TECHNICAL REPORT	DEC 2011 – SEP 2013

4. TITLE AND SUBTITLE

PROBABILISTIC DEVIATION DETECTION AND OPTIMAL THRESHOLDS

5a. CONTRACT NUMBER
FA8750-12-C-0069

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
62788F

6. AUTHOR(S)

Joseph Fahey and Jeff Smith

5d. PROJECT NUMBER
S2AN

5e. TASK NUMBER
PD

5f. WORK UNIT NUMBER
OT

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Bae Systems, Inc.
6 New England Executive Drive
Burlington, MA 01803-5012

8. PERFORMING ORGANIZATION REPORT NUMBER

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

Air Force Research Laboratory/RISC
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11. SPONSOR/MONITOR'S REPORT NUMBER
AFRL-RI-RS-TR-2014-013

12. DISTRIBUTION AVAILABILITY STATEMENT

Approved for Public Release; Distribution Unlimited. PA# 88ABW-2014-0164
Date Cleared: 21 January 2014

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The PDDOT program is to provide a method of continuously monitoring military plans during execution for deviations from expected performance. Determining such deviations in critical to the success of military operations where the complexity of the battle space and rapidly evolving enemy tactics requires an agile and effective response. In addition, commanders face the challenge of deciding when a deviation is significant enough to warrant the risks of re-planning. The PDDOT project identifies an optimal deviation threshold for determining when the executing plan is proceeding as planned, or needs to be modified.

15. SUBJECT TERMS
Plan deviation, optimal threshold, DEEP, StarCraft

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			**KURT LACHEVET**
U	U	U	UU	33	19b. TELEPHONE NUMBER (*Include area code*) **315-330-2896**

Table of Contents

List of Figures

List of Tables

1. SUMMARY

This report describes the research done under the Probabilistic Deviation Detection and Optimal Threshold (PDDOT) project. The goal of the PDDOT program is to provide a method for continuously monitoring military plans during execution for deviations from expected performance as the situation evolves and more information becomes available. In addition, the PDDOT project identifies an optimal deviation threshold for determining when the executing plan is proceeding as planned, or needs to be modified. PDDOT also provides a convenient testing harness for integrating plan execution and deviation parameters into AFRL's Distributed Episodic Exploratory Planning (DEEP) environment to assist development and integration testing. The following sections summarize our approach and results. The remainder of this summary covers:

- Task Objectives
- Technical Challenges
- General Methodology
- Technical Results
- Important Findings and Conclusions
- Implications for Future Research

An introduction to the main body of the report is followed by sections on:

- Methods, Assumptions, and Procedures
- Results and Discussion
- Conclusions, and
- Recommendations and Extensions

1.1 Task Objectives

The task objectives for the PDDOT program are:

1. Develop a method for determining the degree to which a current world state has deviated from a previously planned goal state
2. Monitor an executing plan and world state over time to enable automated plan deviation alerts and re-plan triggers
3. Determine a tradeoff analysis of the reward for re-planning at a particular time

1.2 Technical Challenges

To accomplish these objectives, a range of technical challenges were assessed in this project:

1. Structural representation of state – estimating world state from partial observations in the environment
2. Development of deviation algorithm to identify degree of deviation from the executing plan
3. Integration of the PDDOT agent into the DEEP environment for testing
4. Address excessive thrashing when re-planning occurs too frequently
5. Develop method for identifying optimal threshold

1.3 General Methodology

Our methodology consisted of designing a robust architecture with four major interacting components. First, we addressed the representational challenge of creating a structure to maintain state and actions for the executing plan. Second, a probabilistic deviation detection algorithm was developed to estimate posterior probability distributions over the represented states based on observation data received over time. Third, optimal threshold search techniques were designed and tested to determine the best threshold policies for maximizing reward. Lastly, a set of testing utilities and metrics were designed to test the baseline DEEP system with both no re-planning and PDDOT re-planning techniques.

In addition, parallel to PDDOT design and development efforts, integration with the DEEP system was pursued to provide feedback on both PDDOT and DEEP design and development decisions.

1.4 Technical Results

The PDDOT project was focused on designing and implementing the PDDOT architecture, as well as testing and demonstrating the utility of the overall system concept within the DEEP environment.

1.4.1 Metrics

While the metrics compiled from simulations ran in StarCraft illustrated a functioning PDDOT architecture, further research & development on the simulation platform would be required to fully measure & analyze the performance impact by PDDOT.

1.4.2 Testing Support

The PDDOT project was very successful in providing testing support for both DEEP and PDDOT. A significant number of issues were identified, many of which were also addressed, throughout the development of PDDOT. The testing harness proved to be a very useful tool for customizing and running simulations quickly. In addition, the testing harness also provided very convenient utilities for analyzing test results from simulations within the DEEP environment.

1.5 Important Findings and Conclusions

Testing conducted in the DEEP environment using StarCraft revealed significant deviations from plan expectations and reinforced the need for a PDDOT solution to assist plan execution. In addition, it became apparent that the impact of PDDOT directly depends on the ability to efficiently re-plan to a plan with a greater opportunity for success. In order to provide this capability, DEEP Case-Based Reasoner (CBR) requires a comprehensive case-base in addition to a robust CBR matching algorithm.

1.6 Implications for Further Research

In order to provide a more general solution for deviation analysis in military planning, PDDOT attempts to implement a capability independent of domain specific details from the StarCraft simulation environment used in testing. This goal was achieved through the abstraction of subject matter expertise used to provide incite for the various features comprising state. Interesting research, however, remains to be conducted to better understand the impact of varying feature sets used to perform deviation analysis. In addition, exploration

of feature selection is a potentially useful opportunity in the context of DEEP CBR case matching.

2. INTRODUCTION

The goal of the PDDOT program is to provide a re-planning technology to determine deviations in an executing course of action for real-time management of military operations. Determining such deviations is critical to the success of military operations where the complexity of the battle space and rapidly evolving enemy tactics requires an agile and effective response. In addition, commanders face the challenge of deciding when a deviation is significant enough to warrant the risks of re-planning. Re-planning too often results in excessive thrashing, where plans are not able to setup in time and are primarily occupied with transitioning to the new plan. Infrequent re-planning, on the other hand, eliminates the benefit of dynamic planning and leveraging situational updates to ensure plan goals are achievable. The identification of an optimal threshold between these two extremes is desirable in assisting commanders to determine when to re-plan successfully.

2.1 An Introduction to DEEP

The PDDOT program integrates with AFRL's DEEP environment to simulate military planning and engagements against enemy forces using the StarCraft gaming engine. DEEP is a mixed-initiative decision support system that utilizes past experiences to suggest courses of action for new situations. The DEEP effort began in 2006 in response to deficiencies in course of action development in command and control (C2) systems. The DEEP system aims to provide support to a military planning staff to better satisfy the stated and implied conditions embodied in a given commander's intent. The project uses analogical reasoning over an experience base of past actions to solve new problems. It explores case-based reasoning (CBR) for plan retrieval and course of action development. The system is mixed-initiative in the sense that a commander, through his or her agent, can view and modify the contents of the shared repository as needed. A blackboard-based architecture manages a common knowledge repository through which the various software agents interact.

The resulting initial research platform is comprised of the following components:

- *Distributed Blackboard* to support multi-agent, non-deterministic, opportunistic reasoning
- *Case-Based Reasoning* to capture experiences, which may be successes and/or failures
- *ARPI Core Plan Representation* (CPR) for human-to-machine common dialog
- *Multi-Agent System* for mixed initiative planning

For more information regarding the Distributed Episodic Exploratory Planning (DEEP) project, see Carozzoni, et al. 2008 & Richards, et al. 2012.

2.2 DEEP and StarCraft

StarCraft is a popular military science fiction force-on-force real-time strategy game developed by Blizzard Entertainment in 1998. Blue and red forces build economies, construct physical bases, develop military capabilities and are involved in force-on-force engagements. Three different StarCraft 'races' represent different military forces available to use in force-on-force engagements. Each StarCraft race consists of unique unit and building types with varying strengths and weaknesses when employed on the battlefield.

Figure 1: A screenshot of the StarCraft Brood War videogame

StarCraft is used as the domain for the case-based planning research conducted in the DEEP project. StarCraft was selected for a number of different reasons that make the game engine an attractive choice for DEEP research and testing. One of the primary reasons for using StarCraft is the large case base that can be constructed due to the game's popularity and ability to save simulation logs after game play. Thousands of simulation logs are available online and were collected to create the case-base used in DEEP CBR.

The complexity of the StarCraft gaming engine also makes it an excellent choice for a simulation domain. Although the actions for the red forces in StarCraft are deterministic, the large action space over which the game proceeds makes it unlikely to see the same outcomes given a small set of actions taking place over a limited portion of the game's duration.

In addition, StarCraft is an active research domain for both academic and defense research projects. In 2008, a public domain application programming interface (API), called Brood War API (BWAPI), was developed within the StarCraft research community using C++ to interface with the gaming engine. The framework is free and open source, which helped it gain popularity for researchers, students and hobbyists.

2.3 PDDOT for DEEP

Planning for military operations is notoriously difficult; initial plans rarely survive first contact with the enemy. Initial experiments with the DEEP environment revealed excessive thrashing due to the high frequency of re-planning. It was immediately realized that a challenge existed to provide an automated re-planning capability, including the ability to meaningfully estimate the distance between reality and planned outcomes, and to determine when a deviation is sufficiently significant to justify re-planning. An autonomous and adaptive re-planning

technology provides plausible decisions to commanders aiming to reduce the differential between the current state and the desired outcome.

PDDOT addresses this challenge by leveraging AFRL's in-house research in CBR technology to select the best historical plan for a given situation when determining a re-plan is necessary. In order to achieve this, PDDOT provides algorithms for plan monitoring, representation, probabilistic deviation detection and optimal threshold search. The key components developed under PDDOT are:

- *World State Monitor* for processing situation and plan execution changes over time
- *Deviation Detection* for algorithmically determining deviations in plan execution
- *Optimal Threshold Search* for identifying the optimal deviation threshold
- *Testing Harness* for improving the ability to test and analyze results

The remaining sections of this report discuss the PDDOT deliverables in more detail. Section 3 outlines the methods, assumptions and procedures of PDDOT development. Section 4 includes an overview of each of the four primary PDDOT deliverables, with supporting documentation provided as appendices. Section 5 presents a programmatic summary. Insights and lessons learned for the work developed under this contract are given in Section 6. Recommendations for future work and possible project extensions are given in Section 7.

3. METHODS, ASSUMPTIONS, AND PROCEDURES

Due to the high level of integration between PDDOT and DEEP, a number of requirements were established early in the program to increase both productivity and transparency. The following two sections details the requirements established in order to best achieve success for the PDDOT project.

3.1 Achieving Program Goals

One of the goals for the PDDOT project is to provide testing and integration support for the DEEP environment. Development for the PDDOT project was established in a repository accessible by both BAE Systems and AFRL consisting of the source code for both PDDOT and DEEP for rapid collaboration, development and bug fixing.

In addition, in order for the PDDOT project to provide a useful and robust re-planning service, there are 3 requirements needed from the environment used for integration:

- Real-time, up-to-date situation updates. PDDOT aims to be agile and effective in course of action management, which requires information it reasons about to be available in real-time. In addition, PDDOT's performance also relies on the frequency of situation updates. Infrequent or rare situational updates leaves PDDOT blind to the engagement and hinders its ability to provide insight into course of action management.
- A rich feature-set for both the plan and situation updates to allow comprehensive reasoning over time. Limited information about the state of engagement also hinders PDDOT's ability to gauge deviations in plan execution. A rich feature-set allows PDDOT to provide more insight into the level of deviation for course of action management.
- A mechanism for communicating the need to re-plan when an opportunity to re-plan is identified.

Collaborative development in the shared repository during the early stages of integration testing helped ensure the requirements were addressed by the DEEP environment and utilized in PDDOT.

3.2 Simulation with DEEP and StarCraft

A major goal for the PDDOT project is to provide a useful and robust re-planning service integrated within the DEEP environment and tested with StarCraft. In order to provide this capability, metrics were first established to define success for the PDDOT project. From a simulation standpoint, statistics and scores from StarCraft were considered to provide metrics for PDDOT's performance within the DEEP environment. While StarCraft provides handy metrics to measure the performance of friendly plans executed against the enemy, it is important to note that these metrics involve the entire end-to-end system of PDDOT working within DEEP. In order for these metrics to be a useful measurement of success for PDDOT alone, there are an additional 3 major requirements that need to be satisfied by the environment:

- Reasonable determinism in game-play. In order to have metrics from testing be an accurate representation of PDDOT and DEEP performance, test cases must not vary drastically from one test set to another. Test set sizes are limited due to the amount of time required to run a complete simulation in StarCraft (up to 30+ minutes per simulation – if it terminates), which requires relatively deterministic game-play to have meaningful results with limited test set sizes.
- A rich case-base and intelligent case-base reasoning technology. The concept of leveraging known solutions to solve new ones in case-based reasoning is particularly useful when the case-base of known solutions is rich. However, a case-base lacking relevant cases severely limits the ability to be a useful technology. Case-base reasoning usefulness also declines with naïve querying algorithms, even if a rich case-base is available.
- Comprehensive set of robust metrics for evaluating simulations. The metrics provided by the StarCraft game engine must be relevant to and reflective of the outcome of the engagement.

Implications of the DEEP environment limitations are discussed in the Testing Limitations section (4.4.4), Conclusions (5), as well as the Recommendations and Extensions section (6).

3.3 Enabling Technologies – Deep Green

BAE Systems – Technology Solutions previously led the development of DARPA's Deep Green project and was responsible for developing and implementing a decision-making support system for United States Army commanders planning force-on-force engagements. Under the Deep Green project, a probabilistic model known as a *futures graph* was developed to represent a large, well-defined state space of potential outcomes and utilities for engagements with enemy forces. Deep Green's Crystal Ball component designed algorithms estimating state probabilities and automated alert generation capabilities to enable risk and deviation analysis for commanders. The work accomplished in Crystal Ball provides a strong foundation for adapting existing deviation detection algorithms for the PDDOT program. In order to position the PDDOT program for success, a goal is to consider and leverage the work under Deep Green's Crystal Ball when facing comparable design challenges.

4. RESULTS AND DISCUSSION

4.1 PDDOT Integration with DEEP

Integration with the DEEP environment takes place in two areas, active monitoring during simulation and test management and analysis. During simulation, PDDOT interacts with the DEEP environment through the DEEP Blackboard messaging service.

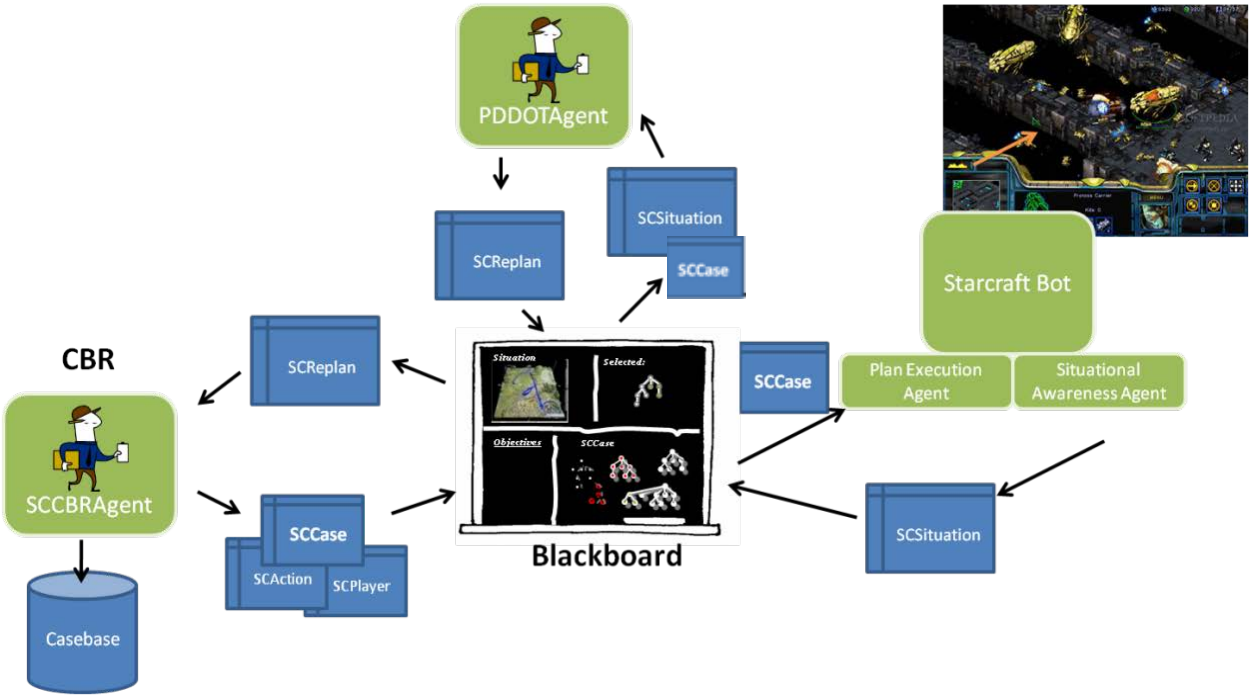


Figure 2: The DEEP Blackboard architecture integrating PDDOT

The DEEP Blackboard represents the common knowledge repository through which agents may interact to conduct non-deterministic, opportunistic reasoning. PDDOT is both a consumer and producer of Blackboard knowledge. In addition to PDDOT, there are 3 other agents involved in contributing and consuming information from the DEEP Blackboard:

> *StarCraft Planning Agent*: Updates the DEEP plan to be executed in StarCraft when notified. Uses DEEP CBR to select a new plan from the DEEP case-base that best matches the current state of engagement when a re-plan has been determined. Posts the plan representation to the DEEP Blackboard.
>
> *Plan Execution Agent*: Executes new plans posted to the DEEP Blackboard in StarCraft. Translates the DEEP plan representation into a series of StarCraft actions over time and executes them.
>
> *Situation Agent*: Posts situational updates to the DEEP Blackboard when state changes are observed in StarCraft. Situation updates include information detailing the creation and destruction of both friendly and enemy buildings and units as well as observations of enemy units by friendly units.

The PDDOT Agent subscribes to the DEEP Blackboard during simulations in order to receive updates about the ongoing engagement from other DEEP agents. PDDOT processes situational and plan updates posted to the blackboard in order to analyze the state of deviation.

In addition, PDDOT posts re-plan updates to the blackboard when deciding a deviation has occurred.

When starting a new simulation in DEEP, PDDOT receives an initial plan update from the StarCraft Planning Agent denoting the first plan being executed. Plan updates are then received only when a re-plan is issued by the PDDOT Agent and processed by the StarCraft Planning Agent. Due to the asynchronous nature of the DEEP Blackboard, plan updates are received after the StarCraft Planning Agent processes the re-plan update, selects a new plan from the case-base and posts the new plan update to the blackboard. On average during simulations, this process occurs very quickly with a delay of less than 50 frames.

Plans used by PDDOT are represented in the DEEP case-base by a series of plan features over time. DEEP plan features for StarCraft plans are comprised of the various unit and building types available by each of the friendly and enemy forces. Plans store a feature-set for each plan time step. Each feature-set maps an integer count to each feature, denoting the number of entities represented by the feature that currently exist on the battlefield. As an example, a feature-set may map the number 6 to Terran Marine, denoting the existence of 6 Terran Marine units in the plan at that particular time step. Currently, plan time steps are defined as 2000 frames in a StarCraft simulation. Typically, plans contain 43 plan states covering a total of 86,000 frames.

Table 1: Terran Units and descriptions used in StarCraft simulations

Terran Unit	Description
Battlecruiser	Largest and most powerful of all Terran air units
Dropship	Terran transport unit capable of transporting any ground unit
Firebat	Infantry support unit equipped with a flamethrower
Ghost	Specialized infantry type unit with the power of invisibility
Goliath	Most powerful anti-air ground unit
Marine	Basic combat unit for the Terran race
Missile Turret	A static turret with basic anti-air defense
Science Vessel	An aerial support unit capable of remotely repairing mechanical units
SCV	Worker unit used to harvest minerals and resources
Siege Tank	A basic tank unit with the ability to inflict high damage
Vulture	A basic mobile ground unit
Wraith	An aerial attack unit with the power of invisibility
Medic	A basic ground unit with the ability to heal biological units
Valkyrie	A small sized spacecraft equipped with rockets

Table 4-1 lists the various units for the Terran race, the primary race used to represent friendly forces in StarCraft simulations with DEEP. Each entity maintains a particular capability that it provides for its race. The remaining two StarCraft races have similar unit and building types with comparable capabilities. DEEP with PDDOT aims to identify and utilize the optimal consistency of each entity for the blue force to best defeat the red force in combat.

Situations used by PDDOT are represented by the Situation Agent in DEEP similar to plan states in the DEEP plan representation. Situation updates are posted by the Situation Agent when changes on the battlefield are observed with a maximum frequency of 50 frames (when no changes are observed, no update is posted). Situations contains updates for each of the friendly and enemy units and buildings, describing the number of each visible on the StarCraft battlefield. Situation updates do not provide complete information about enemy units and buildings and only reveal entity counts for those visible by friendly forces. In the StarCraft domain, this incomplete information availability is known as the "fog of war" and describes the limitation of observing the battlefield only to the extent of friendly units. Imperfect information about the state of engagement plays a critical role in the ability of PDDOT to reason about the state of deviation.

4.2 PDDOT Architecture

PDDOT address the challenge to optimally identify deviation points in plan execution through the interaction of four main components: the PDDOT Agent, World State Monitor, Deviation Detector, and Optimal Threshold Search. Each of the four main PDDOT components provides individual solutions to adaptation, representation, and algorithmic challenges and is optimized for clarity and compatibility. An overview of the PDDOT architecture is shown in Figure 4-2 below.

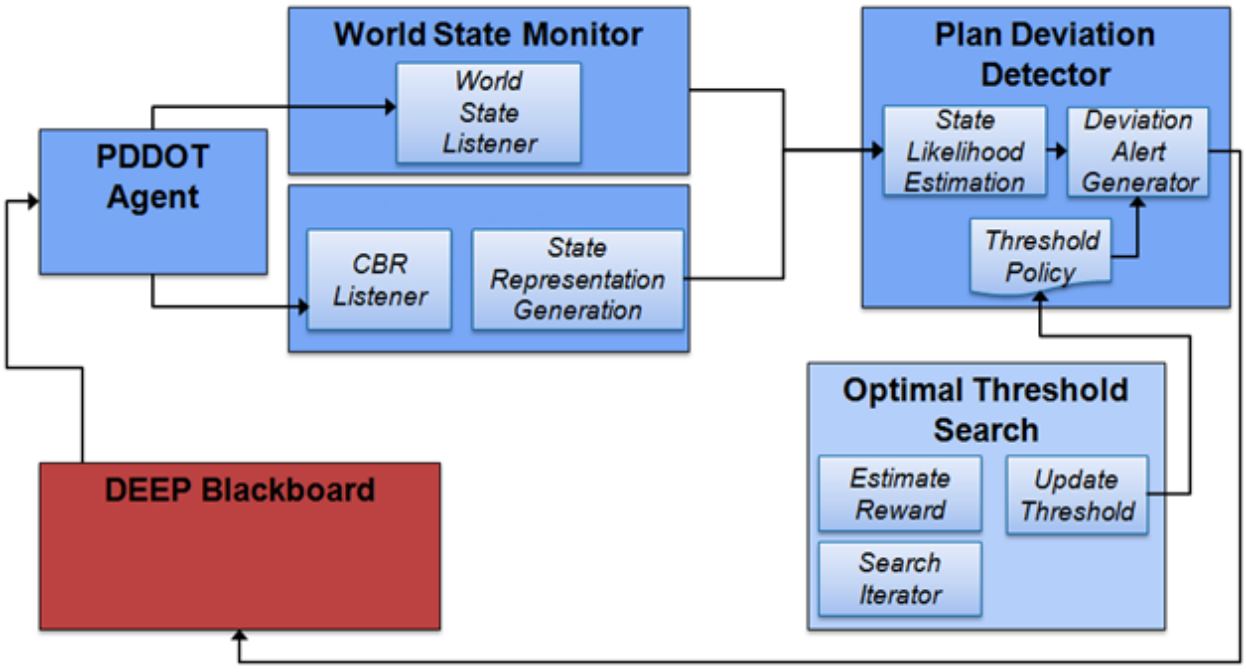


Figure 3: The PDDOT architecture

The PDDOT Agent first addresses the challenge to adapt to the DEEP environment and run tests in the StarCraft domain. Second, the World State Monitor adapts DEEP plan and

situation update representations to PDDOT specific representations in order to effectively analyze deviation likelihoods. Next, the Deviation Detector utilizes a deviation detection algorithm instantiated by subject matter expertise to provide insight on the current state of plan deviation. Lastly, the Optimal Threshold Search component addresses the challenge to discover the optimal deviation threshold to obtain maximum reward.

4.2.1 PDDOT Agent

The PDDOT Agent's primary responsibility is to interface with the DEEP environment and provide a convenient abstraction for integrating the rest of the PDDOT architecture with DEEP. As mentioned earlier, DEEP employs a distributed blackboard system as an opportunistic artificial intelligence application based on the blackboard architectural software engineering paradigm. (Corkill, 1991) The blackboard system functions as a central knowledge store facilitating communication and interaction between the various DEEP agents, including the PDDOT Agent.

The PDDOT Agent uses Java Remote Method Invocation (Java RMI) to interact with the blackboard remotely over a local port connection. Information communicated to the PDDOT Agent from the Blackboard is filtered for plan and situation updates, represented as Java objects in the native DEEP format. Additionally, the PDDOT Agent filters for a specific message posted by the Situation Agent denoting the end of the current simulation. This allows PDDOT to initiate its post-simulation processing to output simulation results. All information filtered from the blackboard by the PDDOT Agent is communicated directly to the World State Monitor to process and translate into representations useful for deviation detection algorithms.

4.2.2 World State Monitor

A key representational challenge that PDDOT faces is the need to analyze plan and situation updates native to the DEEP environment operating in the StarCraft domain. The World State Monitor is responsible for addressing this challenge by constructing state and observation representations useful for the probabilistic reasoning algorithms exercised in deviation detection.

In artificial intelligence research, dynamic models capable of estimating future states under uncertainty are required to support prediction and expected utility calculations. Typically, modeling under uncertainty requires the ability to define qualitative states within a state space, dynamically update likelihood estimations from information disclosed in observations, and determine outcome utility. In order to accomplish this, PDDOT implements a World State Monitor to address the representational challenge of identifying the key variables that constitute qualitative state, as well as the state space of potential states and transitions. The World State Monitor processes plan, situation and termination updates from the PDDOT Agent to provide a graphical model with the necessary information to effectively estimate deviation likelihood during engagements.

One of the reasons BAE Systems – Technology Solutions was uniquely suited to address the challenges of PDDOT was the prior work accomplished under DARPA's Deep Green program for similar ground-based force-on-force planning scenarios. The Deep Green program successfully demonstrated the ability to model pre-defined partial state spaces using a unique hybrid representation of hidden Markov models (HMMs) and partially observable Markov decision processes (POMDPs). The foresight during PDDOT program planning was that the

models, algorithms, and expertise matured under the Deep Green program would translate to the PDDOT domain.

Initially, the probabilistic model considered for PDDOT was a more efficient adaptation of a *futures graph* using a POMDP. In addition to its use in Deep Green, a POMDP is an attractive choice due to the constraint imposed by observations that only partial information is revealed about the state of engagement. While representing all possible states of an engagement is intractable, a simpler representation was considered involving a linear progression of plan states with an additional catch-all deviation state. Each plan state in the POMDP would include a transition to the deviation state with a static, pre-defined deviation likelihood. Similar to Deep Green, a particle filter would be used to estimate the posterior probability distribution over POMDP states based on situation updates.

During the early stages of PDDOT design and integration, it was revealed that the characteristics of the DEEP environment constrained the effectiveness of using the modeling approach taken in Deep Green. Typically, two requirements must be met when considering the use of a POMDP for reasoning –the ability to control state transitions with a transition model detailing the likelihood of action outcomes, and uncertainty about current state. In the Deep Green project, commanders include various decision points in the planning process and utilize futures graph likelihood estimates to provide insight into the outcome of potential actions during plan execution. The use of decision points in Deep Green plans warrants the consideration of a POMDP to optimize decision making for the commander. Unlike the Deep Green project, no decision making is involved in plan execution within the DEEP environment for StarCraft simulations. From the PDDOT perspective, plans are executed independently and observations dictate all transitions between plan states and the deviation state. PDDOT does not *control* the transition between plan states and the deviation state, it *observes* them. The policy of PDDOT is to re-plan when the current state is the deviation state and results in the construction of a new graphic model. In other words, PDDOT controls when the graphic models are replaced, but observes the nature in which they evolve.

Without the need to model decision making in the state space, another model to consider is a hidden Markov model (HMM). While HMMs do not involve the observer controlling state transition, they do include the second requirement of a POMDP in that the current state cannot be directly observed. This particular requirement appears to be relevant to the DEEP domain based on the condition that situation updates only reveal partial information about the entire state of engagement. However, as mentioned earlier, the model considered for PDDOT involves only plan states to avoid the intractability of modeling the entire state space. Taking this approach, uncertainty about current state would imply the possibility of being in different plan states at any given time. Addressing this issue early in the PDDOT program, it was decided that temporal shifts in plan execution would be considered deviations. This decision was made due to the fact that plan execution cannot be augmented based on the likelihood of being in a particular plan state. Instead, if a different plan state better represented the current engagement, re-planning would likely allow DEEP CBR to choose the better state and execute the corresponding build actions.

Ultimately, the state that is most important to PDDOT at a given time is the current executing state. Without the requirements used in POMDPs and HMMs above, the PDDOT plan representation is capable of being modeled as a Markov chain where the current state is either the plan state being executed or the deviation state.

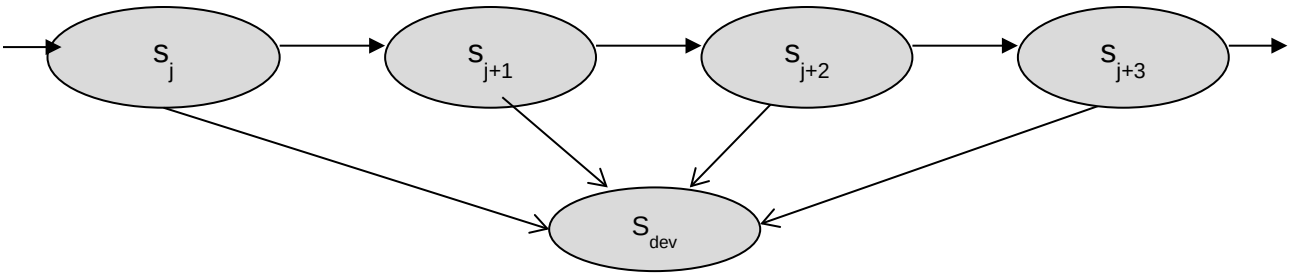


Figure 4: State representation during execution

The World State Monitor is responsible for constructing the PDDOT state representation as a Markov chain from the DEEP representation used in CBR. Each of the PDDOT states are populated with the unit and building counts for both friendly and enemy forces as described in the DEEP case log.

Lastly, the World State Monitor is also responsible for constructing observation updates from the DEEP situation update to allow deviation detection algorithms to appropriately process state transitions in the PDDOT state representation. Observations closely resemble the states representations to allow ease of processing, and include the features necessary to assess state likelihood estimations in deviation detection.

4.2.3 Deviation Detector

The Deviation Detector is responsible for implementing the algorithms that calculate the magnitude in which plan execution has deviated from its expected performance. Deviation detection begins with the PDDOT state representation provided by the World State Monitor, and uses observations also provided by the World State Monitor to determine state transitions.

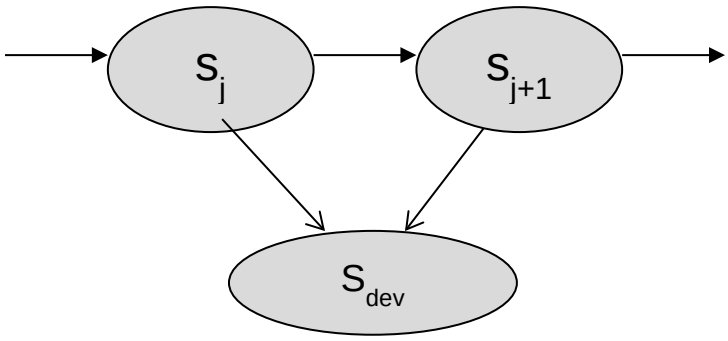


Figure 5: States considered during deviation detection

As described in the World State Monitor, the PDDOT state representation is based on a Markov chain and only considers the executing state and deviation state when determining a deviation. At any specific time during plan execution, the DEEP Plan Execution Agent is actively executing the actions in StarCraft required to transition blue units and buildings from one plan time step to the next. For instance, if one plan time step contains 6 friendly marine units, and the following contains 10, the Plan Execution Agent builds 4 friendly marines over the course of the time step. The PDDOT plan state representing the first plan time step is known as the executing plan state. In deviation detection, the Deviation Detector uses the information described in observations to determine if the executing plan state successfully transitions to the next plan state or results in a transition to the deviation state.

To determine which state transition occurs, observation updates are used to measure how strongly the information conveyed in observation updates represents the executing state transition in comparison to the deviation state. In the context of DEEP and StarCraft, the unit and building counts described in observation updates are compared with those in the state representation using a calculation to generate the strength measurement in terms of likelihood.

PDDOT uses a Bayesian approach to calculating state transition likelihood when observations are received. One challenge that a Bayesian approach faces, however, is the need to calculate the probability of observed parameters outright. This type of calculation is not straight-forward in the context of PDDOT. Nonetheless, a calculation representing the absolute probability of a state transition is not necessarily what PDDOT needs to assess a deviation. A useful calculation that avoids this requirement is assessing the ratio of probabilities for transitioning to plan or deviation state. Conveniently, calculating this ratio allows us to avoid outright probability calculations as they cancel out. In reality, this ratio represents the probability of being on plan to off plan and is perfectly capable of threshold analysis.

The assumption made for the PDDOT project is that all features of the plan state, detailing unit and building counts, are independent. This means, for instance, that the number of marine units present implies nothing about the existence of battlecruiser units. While this may not always be true in practice, an added level of domain expertise is required to determine the nature of conditional independence between various features. Due to program time and resources constraints, this was not prioritized for the StarCraft domain, but is addressed in Section 7 on Recommendations.

$$\frac{P[S|\vec{D}]}{P[\bar{S}|\vec{D}]} = \left(\prod_i \frac{P[D_i|S]}{P[D_i|\bar{S}]}\right)\left(\frac{P[S]}{P[\bar{S}]}\right)$$

Figure 6: Probability calculation of being on plan to off plan

Deviation detection instead assesses the overall likelihood of being in the current state as the product of the likelihood that each observed feature appears to be on plan. Figure 4-5 shows the Bayesian logic involved in calculating the probability of being on plan to off plan with the assumption that features are independent. The second term in the product represents the probability that the executing plan was chosen and is a constant for all calculations. Conveniently, because PDDOT is interested in threshold analysis and not outright value calculation, this term can be ignored in deviation assessment. Deviation analysis focuses on calculating the first term in order to assess deviation likelihood.

PDDOT leverages the inverse functions used in the particle filter for Deep Green's Crystal Ball to calculate the likelihood of the various features being on plan. The inverse function utilizes static maximum and minimum likelihoods to constrain the shape of the functional form and provide reasonable approximations. Two inverse functions are combined, one for calculations involving observations below the planned value and one for observations above the planned value, to accurately measure deviation for arbitrary observations. An example of the functional form is shown in Figure 4-6 below. The shape of the functional form appropriately exhibits the desired behavior to provide high likelihood estimations when observations closely resemble the planned value, decreasing as the observation becomes less accurate.

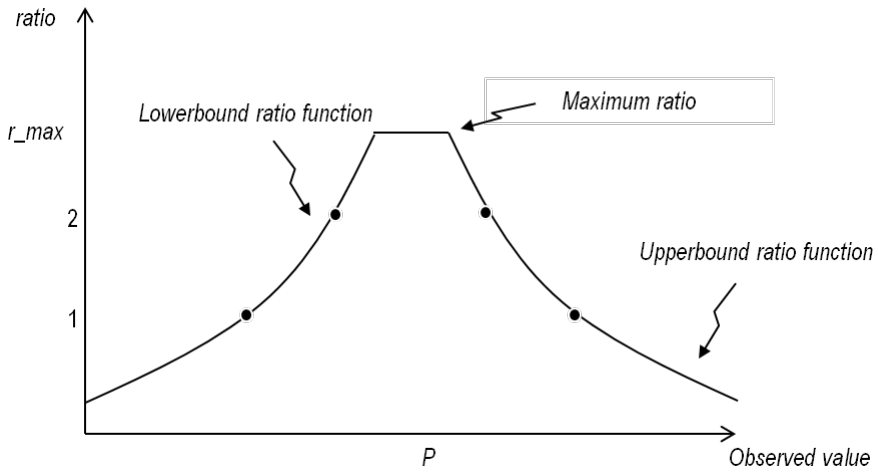


Figure 7: Graphed functional form for calculating likelihood for an observed feature

When developing these calculations, it was quickly realized that a significant understanding of the features involved was required to provide accurate estimates. As an example, an SCV in StarCraft is an inexpensive worker unit used in large numbers. A marine battlecruiser, on the other hand, is one of the most expensive deployable vehicles and is used very sparingly if at all. An observation revealing a few less SCVs than planned would be seen in an entirely different perspective than an observation revealing a few less battlecruisers than planned. Another desire realized during testing was the ability to allow observations to be less accurate over time. Small deviations in expected unit counts become less significant as the number of expected units increases from only a few too many dozens over the course of a simulation. PDDOT attempts to abstract the StarCraft domain knowledge from the deviation detection algorithms to provide a more useful, extensible system that is capable of operating with various cases, datasets, and domains.

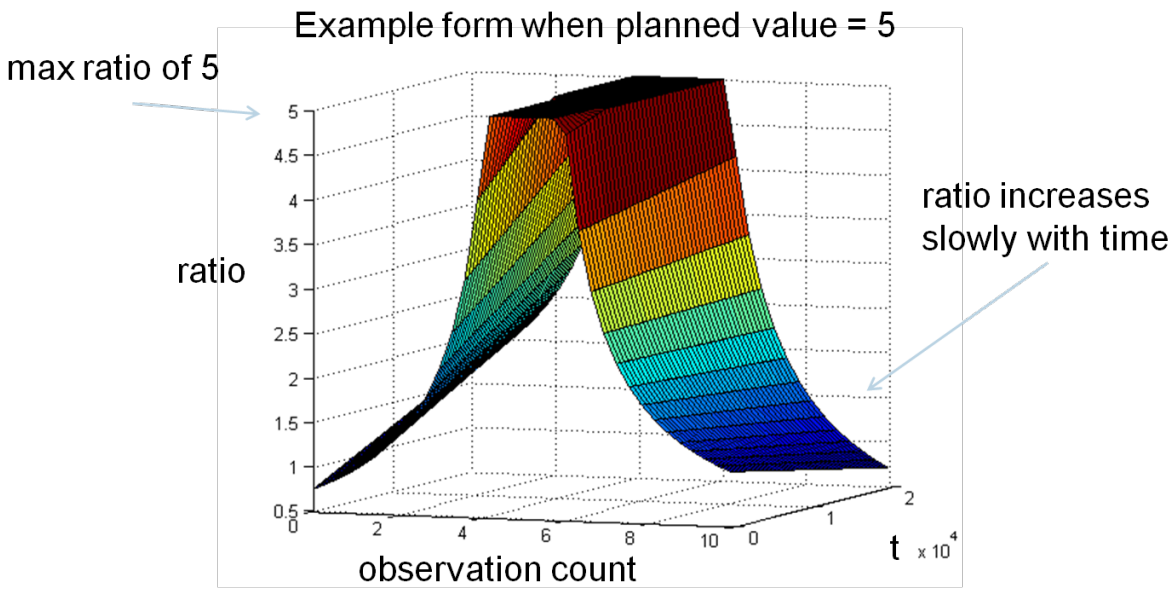
$$P(O_t \mid P_t) = \frac{A}{\%\,plan} + B = \frac{\left[A_1 + (A_2 - A_1)\left(\frac{t - t_1}{t_2 - t_1}\right)\right]}{(O_t - P_t)/P_t} + \left[B_1 + (B_2 - B_1)\left(\frac{t - t_1}{t_2 - t_1}\right)\right]$$

Figure 8: Functional form in deviation detection for features

Deviation detection achieves this abstraction by parameterizing the functional forms used and instantiating them for each feature with the knowledge of subject matter experts (SMEs). Using the knowledge of SMEs, each inverse function is capable of being specifically tailored to a particular feature in a given domain at any given time. The parameterized functional form used is shown in Figure 4-7. A SME worksheet was developed to capture information about each feature in order to instantiate each inverse function with parameters inscribing their knowledge. A sample worksheet eliciting StarCraft SME expertise is shown in Table 4-2. A functional form instantiated with SME expertise is graphed in Matlab in Figure 4-8. The functional form displays the ratio as a function of time and observed valued for the given plan value of 5. Note that the functional form graphed demonstrates the desired characteristic that the ratio increases over time.

Table 2: SME Worksheet Eliciting StarCraft Domain Knowledge

Frame	Ratio elicitation	Marine	Zergling	Vulture	Hydralisk
5,000	1x lowerbound	-0.5	-1.0	-0.5	-1.0
	5x lowerbound	-0.2	-0.25	-0.3	-0.2
	5x upperbound	0.8	0.25	0.8	0.3
	1x upperbound	1.0	0.6	1.0	0.7
50,000	1x lowerbound	-0.33	-0.4	-0.3	-0.5
	5x lowerbound	-0.25	-0.3	-0.2	-0.3
	5x upperbound	1.0	0.2	1.0	0.15
	1x upperbound	2.0	0.3	2.0	0.25



27

Figure 9: Functional Form

Ultimately, the Deviation Detector uses the observations communicated by the World State Monitor to calculate the likelihood of being on plan, and compares the estimate to a threshold to determine if it is significant enough to warrant re-planning. Once a re-plan has been determined, the Deviation Detector creates a re-plan update and notifies the PDDOT Agent to communicate it to the DEEP Blackboard.

4.2.4 Optimal Threshold

While the Deviation Detector is responsible for estimating the magnitude of deviation, another challenge exists in determining the level in which a deviation becomes significant. Early testing in the DEEP environment used the baseline re-planning technique to re-plan on every situational update and resulted in 'thrashing'. The effect of thrashing is severely detrimental and does not allow enough time for a plan to develop, leaving friendly forces defenseless. Figure 4-9 shows the effects of re-planning too frequently and not enough. From the graph, it is apparent that there is some optimal threshold, T*, for which re-planning is the most rewarding for friendly forces. Optimal Threshold Search addresses this challenge by iteratively testing PDDOT at different thresholds and analyzing simulation results.

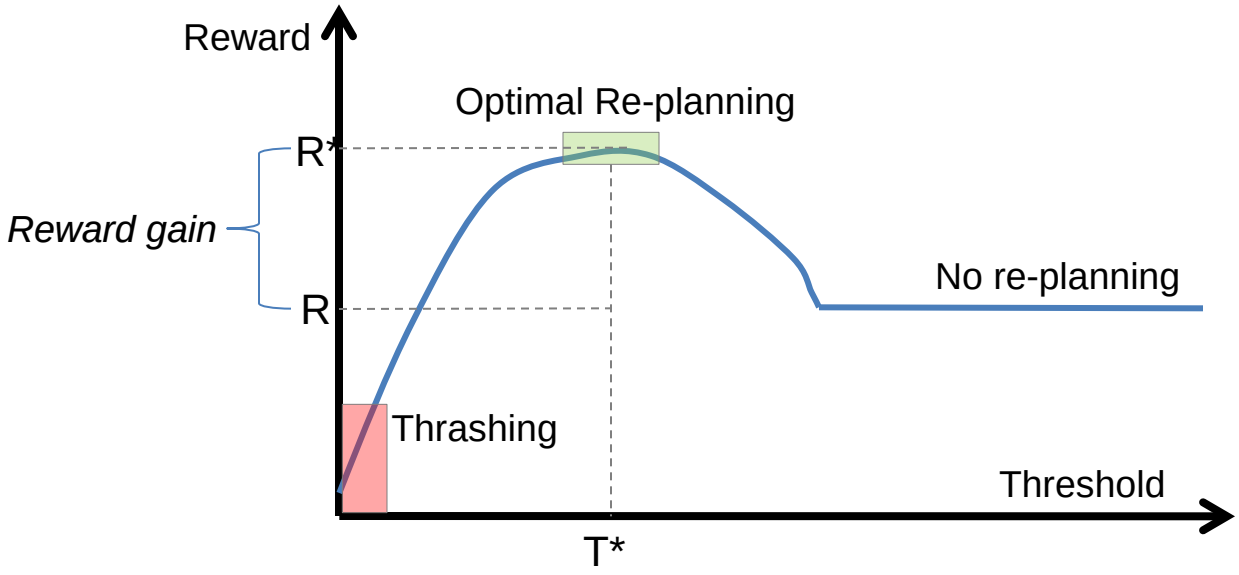


Figure 10: Optimal Threshold Search

Optimal Threshold Search utilizes the heavily parameterized nature of PDDOT to allow the deviation threshold to be dynamically updated during runtime. The test harness (discussed in Section 4.3) developed to provide testing support to both DEEP and PDDOT conveniently handles PDDOT parameterization, including threshold specification.

In order to identify the optimal deviation threshold, Optimal Threshold Search iteratively increments the threshold used in deviation detection by a pre-specified step size. A number of simulations are conducted for each threshold value and the results are analyzed in the test harness to determine the particular threshold that performed the best against the enemy (the number of simulations conducted at each step varied depending on the time available for testing). In addition to supplying a step size for iteration in optimal threshold search, upper and lower bounds are specified to provide a more focused search. The ability to adjust the step

size and bounds greatly improved the process of optimal search, due to the large investment of time required to perform simulations in the DEEP environment using StarCraft. Details regarding optimal threshold search in the test harness is described further in the following section (4.3).

4.3 Testing Harness

In addition to deviation analysis, one of the main efforts of the PDDOT project is to provide convenient testing tools for both PDDOT and the DEEP environment. Early in the program, a large portion of the time spent during the PDDOT design process was understanding the nature of integrating with the DEEP environment. It was critical to fully comprehend the information conveyed in plan and situation updates as well as the manner in which they are delivered to the PDDOT Agent. During this process, a number of bugs were fixed, features added, parameters adjusted, and limitations noted that affected PDDOT assumptions. At the time, the DEEP environment had little to no testing outside of the DEEP developers and relatively limited testing in general.

During the first half of the program, PDDOT work was occasionally put on hold while bugs were fixed or features added to the DEEP environment to address various issues or requirements for PDDOT to operate. In addition, significant time was spent testing and providing feedback for the frequent updates to DEEP source code. Shortly thereafter, an effort was proposed to develop an interface to assist testing and parameterization to improve the ability to test both PDDOT and DEEP as they evolved in parallel over time. The test harness maintains a lasting effect for the DEEP environment as a front-end to testing a development for the DEEP environment for future projects and use at AFRL.

The testing harness was developed within the DEEP source code and integrated with the entire end-to-end system involving PDDOT, DEEP and StarCraft. The main features of the testing harness include the user interface, runtime parameterization for DEEP and PDDOT, as well as simulation results analysis.

4.3.1 Runtime Parameterization

The testing harness developed acts as a front-end to running simulations in the DEEP environment. The harness provides the ability to specify parameters in four major areas: initial conditions, re-planning, duration of testing, and results.

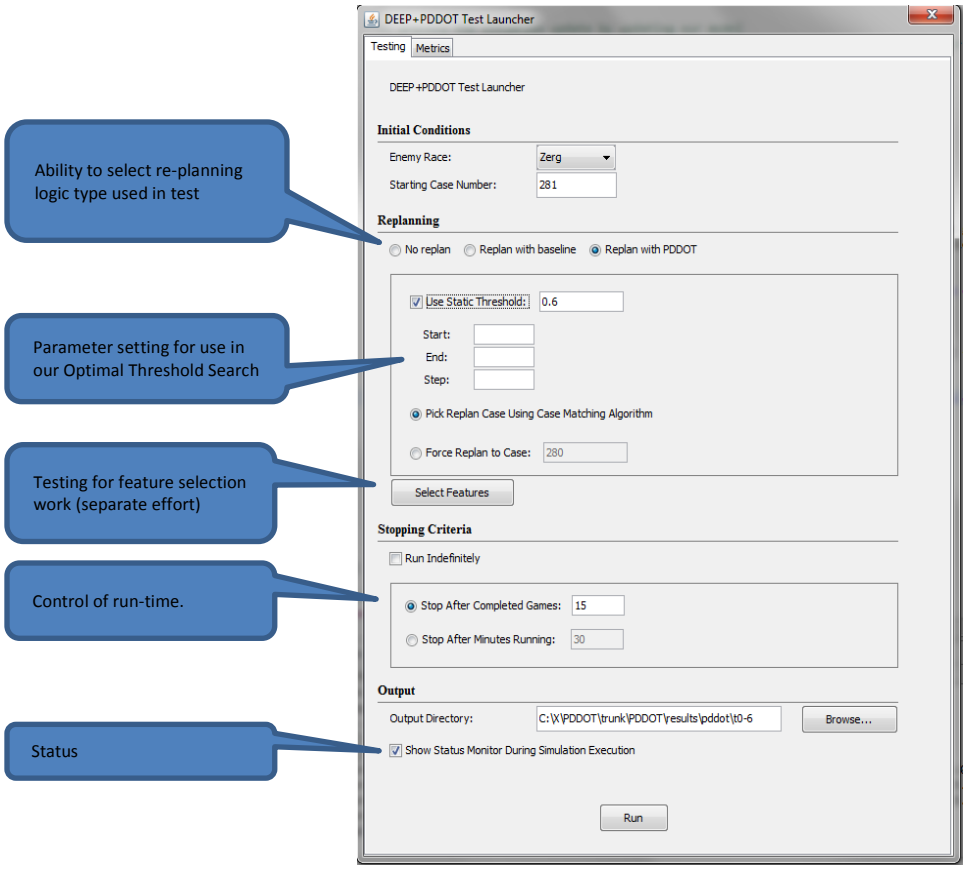


Figure 11: Testing harness for runtime parameterization

During initial testing in the DEEP environment, PDDOT faced challenges simulating force-on-force engagements against the Protoss enemy StarCraft race. A particular characteristic of the Protoss race results in a high percentage of early attack strategies in StarCraft. While it was not understood at the time, the initial case-base plan chosen for the friendly Terran race has extraordinary difficulty defeating early Protoss attacks due to the amount of time it typically takes to construct its defenses. The effect of this match-up severely limited the ability to test PDDOT and let the plan take form. For this particular reason, it was desirable to execute tests against the Zerg race; a race less prone to execute a gimmicky "all-in" early attack. The test harness addresses the effects of varying enemy race and initial case-base plan by allowing the initial conditions to be specified at runtime. The harness modifies StarCraft configuration files and calls DEEP CBR methods to achieve this behavior.

The majority of parameters available for configuration in the test harness fall under the re-planning category. All parameters in this category affect the performance of PDDOT in DEEP. The current harness provides three different methods of re-planning to choose from: no re-planning, baseline re-planning, and PDDOT re-planning. Baseline re-planning refers to the initial DEEP re-planning logic, where a new plan is chosen from CBR at each situation update. When PDDOT re-planning is selected, options become available to adjust the threshold manually, perform optimal threshold search, and even choose a specific plan for re-planning (instead of using DEEP CBR). This option was used for testing early in the PDDOT program when only a small case-base was available for DEEP. Due to the difficulty DEEP

CBR matching had on the small case-base, early PDDOT testing techniques involved enumerating all possible re-planning scenarios, for a single re-plan, and analyzing results.

The 'Select Features' button in the re-planning section, while not functional, serves as a place holder for where feature selection work would integrate with the test harness. As noted in the recommendations section, a very interesting extension to both PDDOT and DEEP CBR is feature selection. Feature selection allows both the CBR and PDDOT algorithms to be biased based on the significance of certain features. The test harness is a useful tool to allow users and developers to adjust weighting and other parameters associated with feature selection at runtime.

Lastly, the test harness provides support for adjusting the duration of testing and results output. Additionally, a checkbox is available to provide a console during simulations to show the status of testing and other testing information. Currently, the status monitor displays the number of simulations completed, the number remaining, and the total time spent testing.

4.3.2 Results Analysis

Results analysis is another important feature included in the testing harness to provide an analysis of the simulations executed by DEEP in StarCraft. The tools provided to analyze testing results was the primary mechanism used to characterize the performance of PDDOT in DEEP. The results analyzed are those captured and outputted by PDDOT in the results handler (see Section 4.4.1). The second tab located at the top of the testing harness reveals a separate user interface for analyzing past results.

Upon selecting the directory containing the results of interest, a list of available output files is displayed. The test harness parses results in XML format and extracts the high level simulation characteristics, such as final score, and constructs a results summary to give the user an overview of the simulation outcome. Underneath the simulation summary, a list of plan states and observation updates is available to the user to further inspect the details of the plan and observations from the simulation. The plan states and observations are interlaced according to time to allow chronological exploration. Lastly, the user can select 'View Scoring Graph' to graph the game score over time and can optionally save the graph as an image file.

The test harness conveniently allows the analysis of either individual or multiple tests at once. The ability to analyze multiple results is a useful feature for PDDOT when attempting to understand the performance of results with similar parameters. For instance, test results for simulations using the same threshold value can be analyzed together to gain an understanding about PDDOT's performance at a particular iteration of optimal threshold testing.

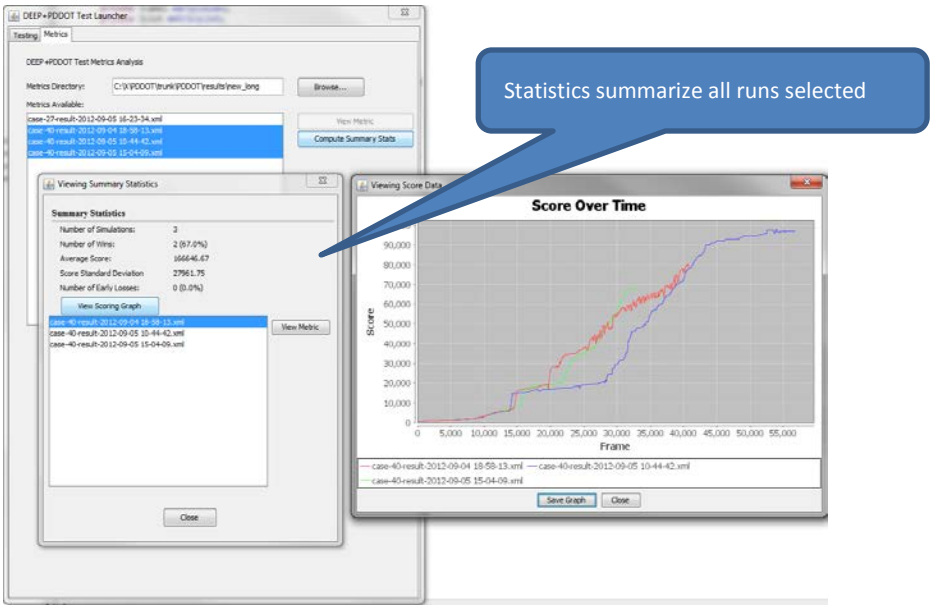


Figure 12: Analyzing Multiple Simulation Results

When selecting multiple result files, the 'Compute Summary Stats' button becomes available. Selecting this button brings up a similar display to the one displaying a single results analysis. The summary section characterizes statistics involving all simulation outcomes, such as average score and number of wins. Instead of displaying a list of plan states and observations, a list of the individual results involved is shown and allows the user to further inspect individual results. Score graphing is also available to display all game scores overlaid on a single graph. A screenshot of the user interface when viewing multiple simulation results is shown in Figure 4-11.

4.4 Test Results

4.4.1 Recording Results

PDDOT includes an implementation of a results handler responsible for recording useful simulation details for later analysis. Simulation results recorded by the results handler is especially useful for debugging PDDOT behavior, analyzing PDDOT performance in DEEP, and generally making simulations more transparent for later analysis.

During a simulation, PDDOT provides the results handler with all runtime parameters, executed plans and observation updates to process and record information for later analysis. For each plan and observation, the results handler records unit and building counts in addition to the time in which they occurred. After a simulation finishes, the results handler captures the final game score and whether or not the enemy was defeated. Lastly, all information recorded by the results handler is outputted as a file in XML format.

4.4.2 Test Plan

PDDOT testing consisted of four separate test sets: no re-planning, PDDOT re-planning, feature exploration, and optimal threshold. The baseline Case-Based Planning uses the minimal amount of information avaialable at the beginning of the game and determines the best possible plan to use with the available information. The baseline does not perform subsequent

re-plans. The PDDOT re-planning test uses the PDDOT architecture & uses information as it becomes available to alert the DEEP case-based re-planning algorithms to retrieve a new plan for execution when the plan is deviating from expected performance. Table 4-3 shows the results for both no re-planning and PDDOT re-planning techniques.

Table 3: Results for testing no re-planning and baseline re-planning

Run Type	Avg Reward – Wins	Avg Reward – Losses	Max Reward	StdDev Reward	% Terran Win
Baseline Case-Base Planning	106,464	23,478	142,865	47,234	10/26 (38.5%)
PDDOT Re-planning	102,076	32,193	199,620	53,384	13/22 (59.1%)

The feature exploration test set tests PDDOT re-planning logic with SME elicitations for individual features. Testing SME judgment for each feature helps ensure the elicitations are correct and provides insight into the impact each feature has on re-planning. Two features were chosen from each race for exploration, and the results for re-planning are listed in Table 4-4.

Table 4: Results for Testing Individual Features Using SME Elicitations

Run Type	Ave Reward – Wins	Ave Reward - Losses	Max Reward	StdDev Reward	% Terran Win
Terran Marines	116,680	21,624	153,110	34,088	4/34 (12%)
Zerg Zerglings	166,045	23,570	166,045	40,915	2/22 (9%)
Terran Vultures	104,605	17,698	104,605	22,712	2/22 (9%)
Zerg Hydralisks	123,200	17,976	124,905	32,783	2/21 (10%)

Re-planning in optimal threshold search included all four of the SME elicitations for Terran Marines, Zerg Zerglings, Terran Vultures and Zerg Hydralisks. While a useful exploration would be to determine which set of features yielded the best results, it would take a considerable amount of time ($2^4 - 4 = 12$ additional test sets) and was decidedly omitted from

the test plan. In addition, due to lack of concretely being able to determine improved performance, testing for an optimal feature set was not a valuable investment of time.

Optimal threshold testing was conducted using the threshold bounds 0.6 to 1.0 with a 0.1 step at each iteration. Tests using the thresholds 0.2 and 2.0 were also conducted as a reference point. The results from optimal threshold testing are shown in Table 4-5.

Table 5: Optimal threshold search test results

Run Type	Ave Reward – Wins	Ave Reward - Losses	Max Reward	StdDev Reward	% Terran Win
0.2	94113	27662	101600	41462	6/74 (8.1%)
0.6	109921	20789	148070	36183	6/46 (13.0%)
0.7	98465	25695	150765	33325	7/71 (9.9%)
0.8	99093	32708	106355	45730	3/32 (9.4%)
0.9	107999	20766	144375	39434	5/42 (12%)
1.0	116113	20252	127935	33488	2/21 (9.5%)
2.0	99880	19117	134130	28101	5/71 (7.0%)

4.4.3 Results Breakdown

Based on the results revealed in Section 4.4.1, the simulations showed that PDDOT had a positive impact on the winning percentages over the baseline planning technique. Although it shows that re-planning out-performed single static plan, it is difficult to conclude that re-planning with PDDOT had a significant impact on the winning percentage over other replanning techniques given the set of tests.

A number of reasons factor into the impact PDDOT was able to provide for DEEP in StarCraft simulations. One consideration is that as the number of tests conducted in each test set approaches a very large (and unrealistic at this stage in development) number, various testing limitations and randomness would approach a constant. We contend that at this limit PDDOT's impact would be apparent. In order to produce better test results, the following section details many of the testing limitations uncovered throughout the program.

4.4.4 Testing Limitations

Throughout PDDOT development and testing a number of limitations were documented. This section provides a list of all limitations identified and is a useful reference for future development and testing in both DEEP and PDDOT

4.4.4.1 StarCraft Domain Analysis

StarCraft is certainly a useful domain for rapidly testing military planning techniques in a force-on-force gaming environment with considerable complexity. While StarCraft has many upsides for accomplishing the goals of DEEP and PDDOT, a number of limitations exist that impact performance:

- Simulations are timely. While StarCraft force-on-force simulations are magnitudes faster and easier than other military planning simulations, it is still a slow process and typically takes anywhere from a few minutes to a half hour to finish (when successfully terminating). The DEEP team has investigated ways of deactivating the user interface allowing much faster simulation runs, however, it has not yet been incorporated into the expirementation platform used in this project.
- StarCraft AI frequently exercises early adversarial attacks. As mentioned earlier, early attacks can be difficult to prevent and don't allow the friendly plan to develop. Due to the large amount of time spent testing, this can slow down the process of generating meaningful results even further.
- Enemy attacks are unpredictable and sudden. Often times, the enemy will send a large number of ground forces to attack friendly bases. When a large fog of war exists (typically early in simulations when much of the map is unoccupied), these attacks have a similar impact to an ambush and only becomes noticeable when the enemy is nearly inside the friendly base. Because of this, re-planning has little impact as friendly forces do not have enough time to react. A consideration that may alleviate this problem is to use more scouting to attempt to identify enemy rushes more timely.
- Metrics identifying a successful engagement are difficult to generate. While game score is an enticing characteristic to relate to success, it occasionally can be misleading. The StarCraft game score combines four individual scores and may not relate to a commander's notion of success. For instance, one of the individual scores included in the game score is a build score – a score that uniformly increases as more friendly units and buildings are built throughout the engagement. When an engagement lasts for a long time, the build score typically drives the game score up. The end result of this is that game score usually increases with simulation time. In reality, a lengthy engagement can be costly and undesirable, whereas more efficient victories are more highly praised.
- Occasionally StarCraft will crash late during a simulation. This can be an issue when friendly victories are somewhat rare and a near victory crashes.

4.5 Programmatic Summary

4.5.1 Milestone Schedule

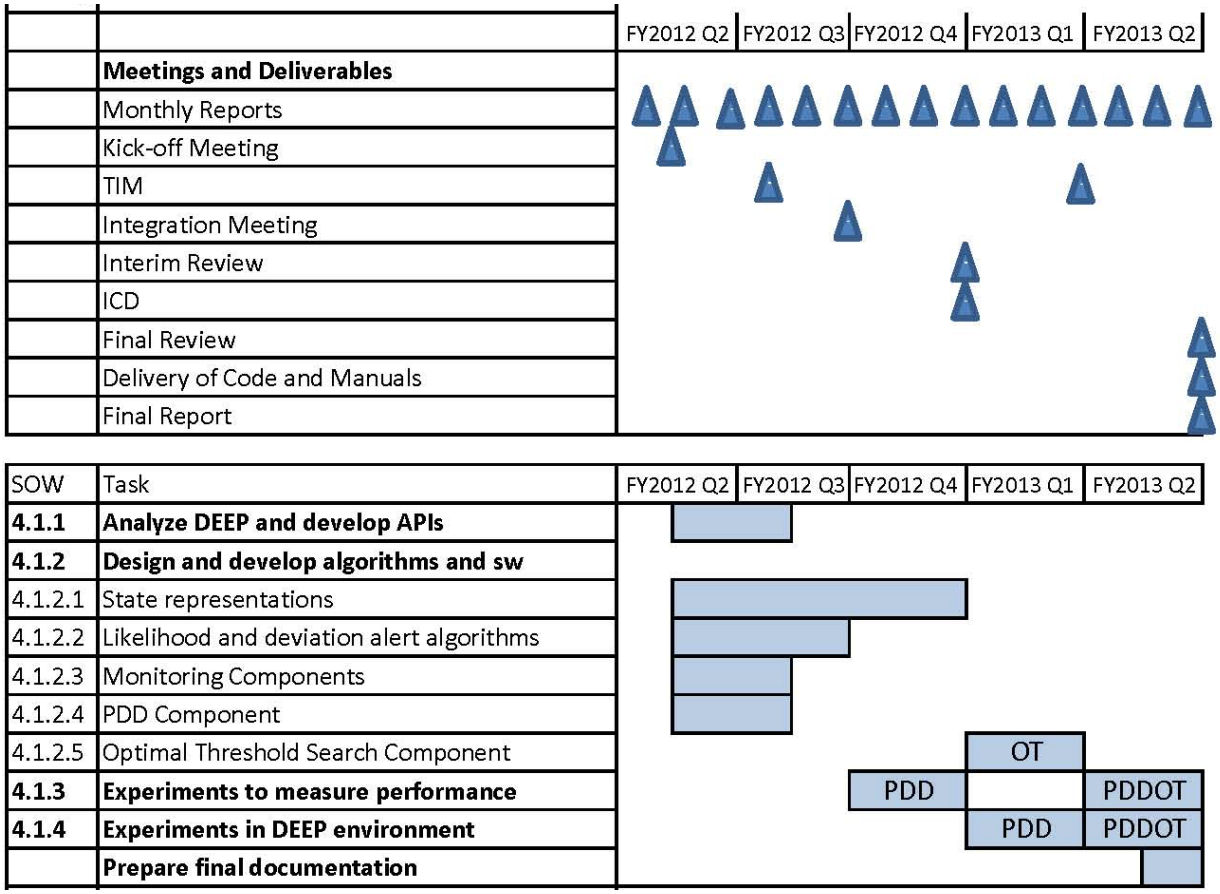


Figure 13: PDDOT milestone schedule

5. CONCLUSIONS

The exploration in deviation analysis accomplished in PDDOT confirms the need to dynamically assess the nature in which plans deviate from expected performance due to battlespace complexity and enemy tactics evolving rapidly over time. Our exploration identifies the challenges that exist in successfully determining deviations during plan execution and provides an architecture and algorithms to efficiently address the challenges involved.

While PDDOT identifies the need for SME expertise for the fielded domain, it achieves domain independence through the use of SME knowledge abstractions to instantiate deviation detection algorithms with careful monitoring. PDDOT's chosen approach allows the architecture to be useful in domains differing from the StarCraft simulation environment targeted in DEEP.

While the DEEP platform with PDDOT witnessed limitations in CBR reasoning and plan execution within the StarCraft domain, it presents a convincing technological approach to having future success in military planning operations with future development and critiques.

6. RECOMMENDATIONS AND EXTENSIONS

We recommend the extension of PDDOT technology from the StarCraft domain to real-world planners by providing an API that may be referenced in a wider context. Towards this end, we had discussions concerning PDDOT's role in AFRL's Living Planner, not only exposing PDDOT's API, but (1) recoded so that it could be provided as a service in AFRL's Cornerstone/SCORA framework (along with several other AFRL-based services including JAGUAR, CDAP/ICAP, etc.) and (2) substituting the blackboard-based DEEP dependency with a common global world state structure. We would be happy to provide the most current architecture diagram representing these discussions. PDDOT's role in a Living Planner is just one instance of participation a real-world planner. PDDOT technology could play a similar role in any dynamic planning application.

Another interesting extension with implications in a more general planning domain is the exploration of feature space for both deviation detection in PDDOT and CBR in DEEP. Both PDDOT and DEEP CBR algorithms operate through reasoning about the various features that comprise plan states and observation updates. Through research conducted to better understand the impact various features have on reasoning, these algorithms can potentially be biased to increase efficiency and accuracy.

A more local recommendation is to extend the DEEP Plan Execution Agent to provide a more realistic, feature-filled capability. In comparison to Deep Green and other planning aids, the Plan Execution Agent might benefit from exploring the inclusion of unit positions and attack coordination and execution. As mentioned in this report, PDDOT currently requires an extremely consice feature set, potentially omitting many of the intricate details that comprise real-world military plans.

Another opportunity exists in using the technology developed by PDDOT to provide a more robust CBR algorithm. DEEP CBR approaches a similar need in determining a plan and state which most closely reflects the current engagement. PDDOT's deviation measurement algorithm could potentially be used in DEEP CBR by searching for a plan state with the lowest deviation measurement. An opportunity exists in determining how the deviation measurement could bridge the gap between DEEP and PDDOT while addressing the efficiency concerns of CBR technology.

As an effort to assist the development of the DEEP environment, a number of limitations were recorded for future consideration. The most noticeable limitations exists in DEEP's case-base and ability of matching algorithm for finding a plan in the case-base that most closely resembles the current state of engagement.

In addition, the following list was compiled throughout PDDOT integration and testing with DEEP:

- DEEP currently uses a default initial build order of buildings and units at the beginning of every engagement. DEEP attempts to alleviate the difficulty in defending early enemy attacks by always creating specific units and buildings before executing the starting plan. While this may have some upside, the initial build order is not currently described in the plan details. From the PDDOT perspective, the initial build order causes the appearance of a deviation and skews the expected unit and building counts.
- The DEEP Plan Execution Agent uses separate AI to control the construction of SCVs and Supply Depots. Similar to the impact resulting from the initial build order,

PDDOT finds it difficult to assess deviation when the expected number of SCVs and Supply Depots differs from the plan and is unknown. In addition, the construction of new SCVs and Supply Depots uses resources that may be required to build other plan specific entities, causing further deviations.

- Marines in bunkers are not accounted for by the DEEP Plan Execution Agent. Observations, however, recognizes the existence of marines in bunkers. Because bunkers with marines are used frequently, a large discrepancy typically arises between the number of marines present and the number recognized by the execution agent. This gives the effect of the plan execution agent building far more marines than the plan states. This also impacts the deviation estimates calculated by PDDOT.
- Plan execution uses a defensive strategy, which often causes stalemates late in engagements. When a friendly engagement with the enemy is nearly victorious, friendly forces may occupy nearly the entire map. However, a small number of enemy worker units may still be lingering in small bases located in the corners of the map, collecting resources indefinitely. In order to declare the engagement a victory and terminate the simulation, the friendly force must eliminate these last few worker units (an easy task, considering they do not have offensive capabilities). This can cause a problem when the plan execution agent doesn't actively search and destroy these last few enemy workers. The effect of this is a stalemate, which will cause the simulation to run indefinitely.
- Occasionally on startup, DEEP fails to initialize Blackboard agents, rendering any agents relying on information produced by the absent agents ineffective.

7. BIBLIOGRAPHY

AFRL/RIS. (2011) *Experience Based Adaption & Re-Planning Simulation Environment: StarCraft.* White paper from DEEP program

AFRL/RIS. (2011) *Experience Based Adaption & Re-Planning Overview.* White paper from DEEP program

Carrozoni, J., Lawton, J., et. al. Distributed Episodic Exploratory Planning (DEEP). Air Force Research Laboratory, 2008. AFRL-RI-RS-TR-2008-279.

Corkill, Daniel; Blackboard Systems. AI Expert 6(9):40-47, 1991.

Hinrichs, T., Forbus, K., de Kleer, J., et al. (2010) *Hybrid Qualitative Simulation of Military Operations.* Technical report from the Deep Green program.

Pearl, J. (2009). *Causality.* Cambridge University Press.

Richards, D., Lachevet, K., et. al. Distributed Episodic Exploratory Planning (DEEP). Air Force Research Laboratory, 2012. AFRL-RI-RS-TR-2012-115.

Stromsten, S. (2010) *Dynamics for filtering/monitoring/tracking in CB* [Crystal Ball]. Technical report for BAE Systems AIT from the Deep Green program.

Surdu, J., Kittka, K. (2008) *The Deep Green Concept.* Proceeding from the 2008 Spring simulation multiconference.

Xu, Tianbing, Zhang, Zhongfei, et. al. (2012) *Generative Models for Evolutionary Clustering.* ACM Transactions on Knowledge Diuscovery from Data.

8. LIST OF SYMBOLS, ABBREVIATIONS, & ACRONYMS

Symbol, Abbreviation, Acronym	Definition
AFRL	Air Force Research Labratory
AI	Artificial Intelligence
API	Application Programming Interface
ARPI	AFRL-Rome Planning Initiative
BWAPI	Brood War Application Programming Interface
C2	Command and Control
CBR	Case-base reasoning
CDAP	COA Development and Analysis Prototype
COA	Course of Action
CPR	Core Plan Representation
DARPA	Defense Advanced Research Projects Agency
DEEP	Distributed Episodic Exploratory Planning
DG	Deep Green
GUI	Graphical User Interface
HMM	Hidden Markov Model
ICAP	Interactive Collaboration Environment for COA Assessment Planning
ICD	Interface Control Document
JAGUAR	Joint Air Ground Unified Adaptive Re-planner
OT	Optimal Threshold
PDD	Plan Deviation Detector
PDDOT	Probabilistic Deviation Detection and Optimal Threshold
POMDP	Partially Observable Markov Decision Process
RMI	Remote Method Invocation
SCORA	Synchronized Constraint-based Optimization, Repair, and Assembly

Symbol, Abbreviation, Acronym	Definition
SCV	Space Construction Vehicle
SME	Subject Matter Expert
SVN	Subversion
TIM	Technical Interchange Meeting
WSM	World State Monitor
XML	Extensible Markup Language