AFRL-IF-RS-TR-2004-75
Final Technical Report
March 2004

REAL-TIME EVALUATION OF CYBER-COURSE OF ACTION (COA) IMPACT ON PERFORMANCE & EFFECTIVENESS

Alphatech, Incorporated

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. J353

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2004-75 has been reviewed and is approved for publication.

APPROVED: /s/

PETER J. ROCCI, JR.
Project Engineer

FOR THE DIRECTOR: /s/

JAMES A. COLLINS, Acting Chief
Information Technology Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	MARCH 2004	Final Jun 01 – Jun 03

4. TITLE AND SUBTITLE
REAL-TIME EVALUATION OF CYBER-COURSE OF ACTION (COA) IMPACT ON PERFORMANCE & EFFECTIVENESS

5. FUNDING NUMBERS
C - F30602-01-C-0169
PE - 62301E
PR - CPAN
TA - 00
WU - 01

6. AUTHOR(S)
John J. Shaw

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Alphatech, Incorporated
525 Brooks Road
Rome New York 13441-4505

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Defense Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington Virginia 22203-1714

AFRL/IFTB
525 Brooks Road
Rome New York 13441-4505

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2004-75

11. SUPPLEMENTARY NOTES
AFRL Project Engineer: Peter J. Rocci/IFTB/(315) 330-4654/ Peter.Rocci@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 Words)***

A fundamental challenge for modern Battle Management/Command, Control, and Communications (BMC3) systems is to withstand attacks against their constituent computer and communication subsystems. However, measures to safeguard or respond to a cyber attack against a BMC3 system, invariably disrupt the processing flow within that system. Thus disruptions may ultimately affect mission effectiveness, and a prudent strategy is to predict those impacts before committing to a specific response or safeguard. The objective of this work is to develop reliable methods for generating cyber Courses of Action (cyber-COAs) in near-real time.

14. SUBJECT TERMS
Battle Management Systems, Command Control and Communications Systems, Computer Attacks, Courses of Action

15. NUMBER OF PAGES
49

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

TABLE OF CONTENTS

LIST OF FIGURES

List of Tables

SECTION 1

INTRODUCTION

1.1 OPERATIONAL SETTING AND MOTIVATING EXAMPLE

A fundamental challenge for modern Battle Management/Command, Control, and Communications (BMC3) systems is to withstand attacks against their constituent computer and communication subsystems. Measures introduced by System Administrators to safeguard or respond to a "cyber attack" will vary with the type of attack, of course, but almost all measures will disrupt the processing flow within the BMC3 system in some manner. Encryption, for instance, will invariably introduce processing and communication delays within the BMC3 system. Re-locating a database server will both introduce delays and interrupt processing. Every BMC3 system is in a race against time, and disruptions in its processing flow may very well cause the system to lose the race.

System Administrators disrupt BMC3 processing flows with startling regularity even under benign conditions when their systems are not under attack, and the consequences are often dire. Two examples from Joint Experiment Force Exercise (JEFX) 2000 illustrate the impact of even "modest" disruption on time-critical BMC3 processing.

In the first example, the Combined Air Operations Center (CAOC) at Hurlburt Field was performing Time-Critical Targeting (TCT) with the TCT cell at Nellis AFB. System Administrators at Hurlburt discovered a minor configuration problem, and attempted to fix the problem with a system upgrade. The upgrade, however, caused a loss of synchronization in the data appearing on the Common Operational Pictures (COPs) at Hurlburt and Nellis. With this loss of synchronicity the CAOC at Hurlburt was no longer able to see targets nominated for engagement by the TCT cell at Nellis. Confusion quickly settled in: the CAOC observed strike aircraft diverted (to engage TCTs) without knowing why. CAOC operators assumed that the air plan had gone awry, and suspended missions in order to straighten out the situation.

The second example involved planned system outage for maintenance at the CAOC. The directors for Current Operations and for System Administration agreed to a plan to take the system down a) after 7PM, and b) after air planners had finished building the strike package. The System Administration staff were instructed on the first condition, but not the second. The system was taken down for maintenance at 7 PM before the air plan was completed, and several hours of data were lost.

In both cases, the System Administrators did not know that their BMC3 systems were in the midst of critical workflows that simply could not be interrupted. This point is vitally important as we look beyond the relatively benign conditions in those circumstances to the challenges of responding to cyber attacks. System Administrators cannot respond intelligently to a cyber attack if they are unable to anticipate how disruptions in the ongoing workflow will affect the mission.

If the impacts of disruptions are understood, then System Administrators can further evaluate the effectiveness of courses of action (CoAs) undertaken to mitigate and to circumvent the effects of cyber-attacks. Reconfigurations of the BMC3 network, whether physical, logical, or procedural, introduce disruptions; however, these short term disruptions may improve performance over the duration of the cyber-attack effects.

The Effects-Based Cyber-COA Optimization Technology & Experiments (EBCOTE) project has studied the problem of quality of service (QoS) assurance in BMC3 systems in the context of a Time Critical Targeting (TCT) cell scenario. In an Air Force Air Operations Center (AOC), the TCT cell is responsible for the command and control of air operations that prosecute time-sensitive targets of opportunity. The TCT Cell must identify and classify emerging targets based on incoming sensor and Intelligence, Reconnaissance, Surveillance (ISR) data, evaluate the targets, and ultimately task surface or air assets to prosecute the targets. In order to successfully prosecute a TCT, operators must use a variety of IT capabilities both hardware and software, to support each of a sequence of doctrinally prescribed tasks.

Since TCTs may be lost to sensors after emergence, there is a short time limit (on the order of ten minutes) for their prosecution. In addition, TCTs may emerge frequently; dozens may be in process at once. Together, these characteristics make the TCT cell's mission effectiveness highly sensitive to disruptions in workflow.

1.2 PROBLEM CHARACTERIZATION

To formalize the problem of QoS assurance in BMC3 systems, we draw from managed information systems ideas in the business process re-engineering (BPR) community, in particular, the notions of a workflow system and workflow management. BMC3 systems that we are studying and their missions are characterized by their support of a doctrinal *process*—tasks to be performed, conditions for executing these tasks, and further subprocesses. A *workflow* is the process-related collection of jobs, resources, and orders to initiate or perform tasks on jobs [1]. The BMC3 information system, then, is a *workflow system*—it supports the workflow to accomplish its mission. Some BMC3 systems also incorporate a *workflow management system* (WfMS), which initiates the execution of activities and tracks the generation of activities and their assignment to resources, the status of jobs and activities, performance statistics, and other data. The measure of mission effectiveness in workflow systems is a quality of service (QoS) metric. The QoS metric defines an optimization criterion based or observable, derived, or predicted parameters of collected workflow data [15].

For BMC3 systems, both cyber-attacks and CoAs are modifications (or reconfigurations) to the nominal workflow system that affects QoS; in EBCOTE, these modifications are to availability and allocation of cyber-resources, although one can formulate more general possibilities. A cyber-resource can be a hardware device, software applications running on that device, data files or databases, or connectivity to any of these. EBCOTE assumes that other software entities in the system can recognize cyber-attacks and categorize them in terms of workflow modification; see Section 2.2 on other DARPA Cyber-Panel projects for intrusion detection and situation awareness.

A workflow model defines a process and the resources that support it. Colored Petri Nets (CPNs), with their concepts of transitions (tasks), queues (states), and tokens (jobs), are commonly used to model workflows. In EBCOTE, we use the CPN model to address a stochastic discrete optimization problem:

> Find allocations of resources to tasks that maintain the best QoS objective score in the face of constraints resulting from cyber-attack disturbances to nominal workflow.

1.3 TECHNICAL CHALLENGES

The EBCOTE project faced several technical challenges in addressing the BMC3 cyber-CoA problem:

1) BMC3 systems exhibit diversity in their workflow models. The EBCOTE solution must be able to flexibly represent this diversity of models and formulate the appropriate optimization problem at run-time.

2) The optimization problem is a complex multi-step queuing problem. Jobs enter the system and wait at a sequence of stations for processing. Value is collected only when a job is processed by the last station. Resources are allocated to the stations to service the jobs, subject to constraints on which resources are capable of supporting which stations.

3) Real BMC3 systems have sophisticated probability distributions for arrival and transition times.

4) The problem is a combinatorial search over possible network configurations, but similar configurations may give very different QoS scores.

Because of these considerations, we cannot practically generate a closed form for a QoS objective as a function of resource allocations using mathematical analysis.

We describe the research context for EBCOTE-3 in Section 2, including related outside research, connections to other DARPA Cyber-Panel efforts, and the foundational results of the first two phases of EBCOTE. Section 3 describes the EBCOTE system architecture; EBCOTE system components are described in detail in the following sections. Important workflow concepts and a driving example are discussed in Section 4, followed by our resource policy and resource configuration models in Section 5. In Section 6, we state the QoS objective optimized in EBCOTE and a description of an oracle for this objective. Our formulation of the central QoS optimization problem and the search algorithm incorporated in EBCOTE-3 are found in Section 7.

SECTION 2

PREVIOUS RESEARCH

2.1 PREVIOUS OUTSIDE RESEARCH

QoS management is a growing research area for distributed or networked IT systems. Building on dynamic resource reservation or allocation in non-workflow contexts such as scientific computing [16] and web multicast [10], the DARPA Quorum program, the predecessor to Cyber Panel, sought to apply QoS management ideas to emerging problems in chains of applications and communications services [17].

Most projects in Quorum investigated the dynamic allocation of computing resources to incoming jobs, e.g., Honeywell's RT-ARM [19]. Although some computing resources are easily allocated in a reactive fashion, cyber-resource assignment in a BMC3 system cannot be completely dynamic. In some cases, if a resource has been assigned to support one kind of task, there is a cost of assignment or reassignment to other tasks, e.g., time for physical reconfiguration or software installation. Such costs motivate a static resource allocation policy that assigns classes of resource tokens to support kinds of tasks, with infrequent movements of tokens between classes. This approach is similar to the Weapons Systems Open Architectures (WSOA) approach. WSOA ideas on managing airborne computing resources have evolved into workflow management requirements in the Multi-Sensor Command and Control Aircraft (MC2A) program.

Recent work in QoS management has focused on integration with evolving WfMS concepts and technology. Cordoso and Sheth [6] identify the need for WfMS mechanisms to model multiple QoS criteria, log historical QoS metric performance, and estimate QoS on a task-by-task basis. Hewlett Packard's Chang*e*ngine [9] is one of the first WfMS-compatible systems for integrating pre-existing resource management systems through principled resource policy specifications.

Analytical work in workflow QoS optimization has centered around providing optimal solutions for deterministic criteria such as system latency. Notable results have included the resource scheduling work of Alhusaini and Prasanna [2], which minimizes latency over a directed acyclic graph workflow formulation for a simple resource model consisting of compute resources and data repository resources. Gertphol, et al, [11] have used integer programming to map applications onto processing paths, once again minimizing system latency.

EBCOTE is breaking new ground in its explicit consideration of QoS optimization in the context of a broad class of workflows with a stochastic QoS metric depending on job completions. This stochastic discrete optimization problem is a challenging one, and we draw on a foundation of recent work in hybrid gradient-based/random search methods which utilize simulation or other means to establish search heuristics. A review of these methods can be found in Andradottir [5]. Our approach

falls into her Perturbation Analysis class, but we extend perturbation analysis to gradient-estimation for search over discrete parameters.

2.2 CONNECTIONS WITH OTHER DARPA CYBER-PANEL WORK

EBCOTE is funded by the DARPA Cyber Panel program. Cyber Panel's program concept is to provide for the protection of mission-critical information systems from cyber-attack. Cyber Panel projects range from simple intrusion detection and event generation, correlation and situation assessment, to threat analysis and prediction.

BBN Technologies' CMIT and Stottler-Henke's Propheteer are systems that are primarily responsible for situation awareness—identifying intrusions and detecting the intent and extent of an attack. These systems do provide CoAs based on a rule-based framework, but do not use optimization to devise effective and novel responses. Situation awareness systems identify the nature of cyber-attacks and could supply key inputs to EBCOTE.

Systems such as Honeywell's CIRCADIA and MASC attempt to resist cyber-attacks in progress by devising reactive, defensive responses that are, like EBCOTE, based on control-theoretic ideas. ALPHATECH's AlphaLADS identifies novel "worm" attacks in progress and determines countermeasures for isolating and defeating the infection. EBCOTE, on the other hand, proposes responses to mitigate impacts of attacks that have already occurred—given that some attacks will succeed, it focuses on repair and recovery of system effectiveness.

2.3 RESULTS OF PAST EBCOTE PHASES

EBCOTE is now coming to the end of its third phase of development, building on work begun in June, 2001. Figure 1 charts the evolution of EBCOTE throughout its three phases.

In our feasibility study for EBCOTE-1, we developed off-line analysis methods for predicting the impacts of cyber-attack disruptions beforehand. We developed an analysis method based on perturbation analysis. The method approximates the nominal BMC3 workflow processing using a Markov model, and computes important sensitivity metrics from Optimal Control Theory, most notably *cost-to-go*, and *co-state*. The key parameters required by this analysis method are Completion State transition probabilities, which might be indirectly derived using data that is regularly collected by advanced workflow management systems.

The logical next step for EBCOTE was to develop an on-line capability to predict the mission impact of candidate Cyber-COAs to respond to cyber attacks. This capability encompasses two elements: an element that predicts how a Cyber-COA disrupts BMC3 workflow, and an element that predicts how a disruption in BMC3 workflow affects QoS. In EBCOTE-1 we modeled workflow in terms of Markov chains, and developed a mission impact prediction method specifically suited to Markov chains. This method proved to be quite accurate, but required considerable calibration to estimate the several hundred state transition probabilities required by the Markov chains. We concluded that EBCOTE-1 was not suitable for on-line use. EBCOTE-2 modeled BMC3 workflow using Colored Petri Nets (CPNs), and

implemented a perturbation analysis method developed for CPNs to predict the mission impact of a cyber-COA; this was the path reconstruction analysis (PRA) algorithm.

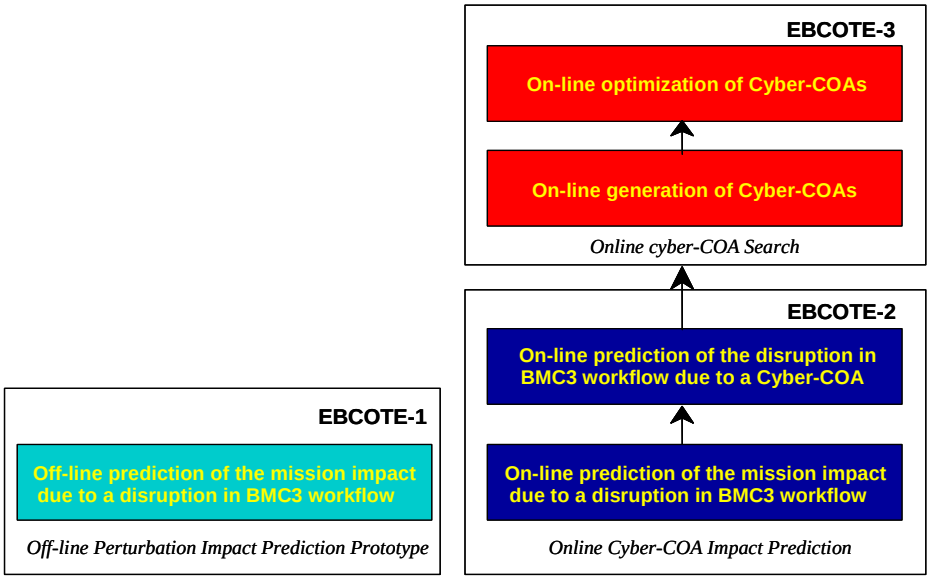

Figure 1. Three research stages in the evolution of the EBCOTE system.

2.4 EBCOTE-3 PROJECT GOALS

The EBCOTE project had four main goals during its third phase:

1) Develop an on-line capability to generate cyber-COAs in response to a cyber-attack.
2) Leverage path reconstruction analysis capability to search for an optimal cyber-COA.
3) Implement a prototype whose architecture enables transition to a WfMC-compliant workflow management environment.
4) Demonstrate ability to generate cyber-COAs in real-time using BMC3 workflow data.

EBCOTE-3 builds on previous work by using the on-line cyber-attack/cyber-CoA perturbation impact assessment capability to determine which cyber-resources are over- and under-utilized. It then identifies prospective activity-resource re-assignments using this heuristic information to guide optimization.

SECTION 3

EBCOTE-3 SYSTEM CONCEPT AND ARCHITECTURE

The EBCOTE system concept, shown in Figure 2, is based on workflow analysis. As such, it leverages data about BMC3 workflows archived in a workflow management system to estimate the effects of perturbations to the workflow and identify ways of reconfiguring the network's cyber-resources to mitigate these effects. EBCOTE-3 assumes that an intrusion detection and assessment tool such as CMIT or Propheteer, also developed under DARPA Cyber Panel, can identify a cyber-attack in terms of a workflow perturbation. These tools are shown notionally in the diagram below, though they are not necessary to run the EBCOTE-3 prototype.

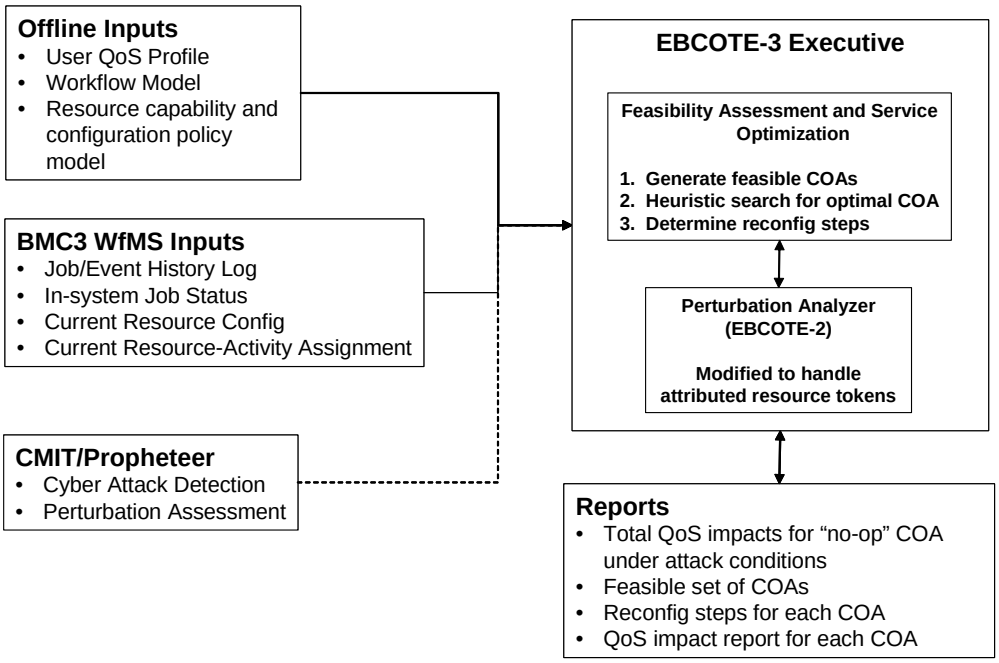

Figure 2. System architecture for EBCOTE-3.

Because the EBCOTE project did not have access to a WfMS-instrumented BMC3 system, we used an ALPHA/Sim Petri Net simulation of the TCT Cell BMC3 system to drive research and development. Our prototype's design, however, permits extension to use of WfMS data from WfMC-compliant systems.

3.1 WORKFLOW MODEL

The workflow model is a topology of queues, queue decision rules, transitions, and transition timing data that specifies a BMC3 workflow process.

A mature WfMS captures this information as a part of its workflow definition process. The EBCOTE-3 prototype imports a workflow model created within ALPHA/Sim to support the path reconstruction analysis (PRA) algorithm.

The prototype can be evolved to import workflow model data using WfMC standards. See the CoA Search SDK Developer's Guide for details on EBCOTE-3's API.

3.2 JOB HISTORY LOG

The log of the processing history of past jobs provides an essential input to the path reconstruction analysis (PRA) algorithm. The log records event information including job ID, activity completed, time of completion, and processing time (delay). The log records attributes of the resource used to support the activity, and event-caused changes to job attributes which are used to evaluate decision rules in the workflow.

A mature WfMS captures this information during prior workflow instances. EBCOTE-3, like EBCOTE-2, generates a flat-text event log using an ALPHA/Sim simulation of a BMC3 workflow.

The prototype can be evolved to import WfMC-standard job history log data using our API design.

3.3 IN PROGRESS JOBS

This input specifies the completion status of jobs currently in the system and the availability status of resources which might currently be engaged in processing activities.

A mature WfMS captures this information while jobs are in the workflow. EBCOTE-3 uses a flat text input file in-progress jobs, but we have also designed an API for querying this information from a WfMS.

3.4 USER/TASK QOS PROFILE

This input is a record of QoS objectives and priorities for specific users and tasks. Cardoso, et al, describe four categories of QoS metrics, including time, cost, reliability, and fidelity [7]. We focus on time objectives (e.g., relative and absolute deadlines) in EBCOTE-3. The profile is used to assign priorities for meeting objectives to certain classes of jobs, and provides weightings for the objective function used to determine COAs that optimize QoS.

A mature WfMS captures this information as a part of its workflow definition process.

3.5 RESOURCE POLICY MODEL

This model describes types of resource tokens that support BMC3 workflow activities—their attributes, domains of valid assignments to those attributes, actual values assigned, and their roles. It lays out qualification policies matching roles to BMC3 tasks. This model also describes valid installations for changing tokens of one resource token type into another type. The resource capability model should not be confused with the nominal resource configuration data which records the actual policy for a workflow instance—activations of resources and their assignment to support tasks.

Workflow management architects have recently developed resource classification and management tools that integrate with a WfMS [13]. These tools capture an organizational model of resources and roles, along with qualification, requirement, and substitution policies. The resource policy model we developed for EBCOTE-3 draws upon these approaches.

3.6 RESOURCE CONFIGURATION

Cyber-COAs considered during EBCOTE-3 reflect changes in activity-resource assignments from the nominal case. Such activity-resource assignment changes can model both physical and procedural network configuration changes. The nominal resource configuration records both the active cyber-assets and their assignments to workflow tasks at the time a cyber-attack occurs. The nominal resource configuration becomes the initial solution used as a starting point for the cyber-COA Search (Section 3.7.2).

Cyber-attacks and Cyber-COAs are modeled as alternative configurations. These configurations reflect both alternative assignments of cyber-assets to tasks and installations modifying the type of a resource token (from the nominal). In any configuration, activities need not be assigned a resource, but this may prohibit completing any jobs, depending on the topology of the workflow process. Resource tokens need not be assigned to an activity. Each potential solution to the discrete stochastic optimization problem is expressed as a system reconfiguration.

BMC3 systems, even those without WfMS instrumentation, are usually able to determine what resources are currently activated and assigned to workflow activities.

3.7 EBCOTE-3 EXECUTIVE

3.7.1 Perturbation Analyzer (PA)

This component assesses the impact of perturbations and resource reconfigurations on the user-specified QoS metric (Section 3.4). The PA uses the job history log to perform path reconstruction analysis, generating new histories by propagating the perturbation along the sample history. PA thus functions as an **oracle** for the discrete stochastic optimization problem—in the absence of a closed form expression for the objective function, the PA allows us to get a QoS value for each potential reconfiguration solution.

The EBCOTE-3 PA capability builds on the work performed in EBCOTE-2. We expanded our workflow model and PRA algorithms developed in EBCOTE-2 to incorporate resources, job classes, in progress jobs, and installation delays.

3.7.2 Cyber-Course of Action Search (COA Search)

This component determines a feasible, near-optimal cyber-COA through search. The PA is used to generate evaluations of the objective function for various solutions, which are in turn used to estimate a gradient for search through the solution space.

The EBCOTE problem is a difficult combinatorial optimization with many local minima. To facilitate “just-in-time” solutions, COA Search reports the best COAs found to date during search, and

restarts using a predetermined number of initial solutions, increasing the likelihood that a better local optimum is found over time.

3.8 REPORT GENERATION

EBCOTE-3 generates an impact report for each cyber-COA considered, including QoS and job success rate by job class. It also reports the details of each configuration option, including activation and installation steps relative to the immediate post-attack configuration.

SECTION 4

WORKFLOW MODELS

Workflow concepts are an essential part of the EBCOTE solution. A workflow model provides an analytical formalism for characterizing the way networks support BMC3 missions and makes a quantitative determination of QoS possible under varied conditions. Petri Nets [1,3] are the preferred methodology for modeling workflows; in this section we discuss workflow concepts and relate them to Petri Net modeling. We reserve a discussion of our resource model to Section 5.

4.1 WORKFLOW MODEL DEFINITIONS

The definitions of workflow terms in this section are adapted from the Workflow Management Coalition's (WfMC) glossary [18].

- *Workflow definition*: The formal representation of a business process which supports automated manipulation. This representation includes a network of tasks performed on jobs, criteria for starting and stopping the process and activities within it, and resources participating in those activities. For EBCOTE, the workflow definition is the Petri Net model itself.
- *Workflow instance (job):* An individual enactment of the process. In EBCOTE, only completed jobs contribute positively to mission effectiveness (QoS). Jobs are represented by Petri Net *tokens*.
- *Task:* A stage of processing in the workflow which jobs pass through on their way to completion. Tasks are represented by Petri Net *transitions*.
- *Task instance (activity)*: An activity is an instance of a task executed on some job. Activities are not explicitly represented in Petri Nets, but each firing of a Petri Net transition is an activity.
- *Process state:* A representation of the internal state of a job in the process. The completion of activities may modify the process state of a job. Process states are Petri Net *queues*.
- *Transition conditions:* A logical expression evaluated by a workflow engine to decide the sequence of task execution within a process. In EBCOTE, these conditions are decision rules at queues in a Petri Net.
- *Workflow relevant data:* Data used by a WfMS to evaluate transition conditions. These are Petri Net *colors* or *token attributes*.
- *Event:* An occurrence of a particular condition (called a trigger) which causes the WfMS to take actions such as process or activity initiation. For EBCOTE, events occur when transition conditions are met such that an activity may commence. For example, when a Petri Net transition fires, this is an event.

- *Audit Data:* The historical record of a process instance from start to completion, normally including information about transition events. This data forms the *Job History Log* on which the Perturbation Analyzer performs path reconstruction to analyze impacts on mission effectiveness.

4.2 EXAMPLE WORKFLOW MODEL

To illustrate workflow models as used in EBCOTE, we describe an ALPHA/Sim Petri Net model of the TCT Cell workflow described in Section 1.1. The top-level view of the TCT process is shown graphically in Figure 3.

Each target candidate entering the TCT "kill chain" is a job to be completed. Completion requires evaluating each target candidate and the possibly assigning air or ground assets for attack. Full value for target processing is received only if air or ground assets are tasked within five minutes of the candidate entering the system; value received decays after that. After ten minutes, the candidate is dropped from processing, and no value is received.

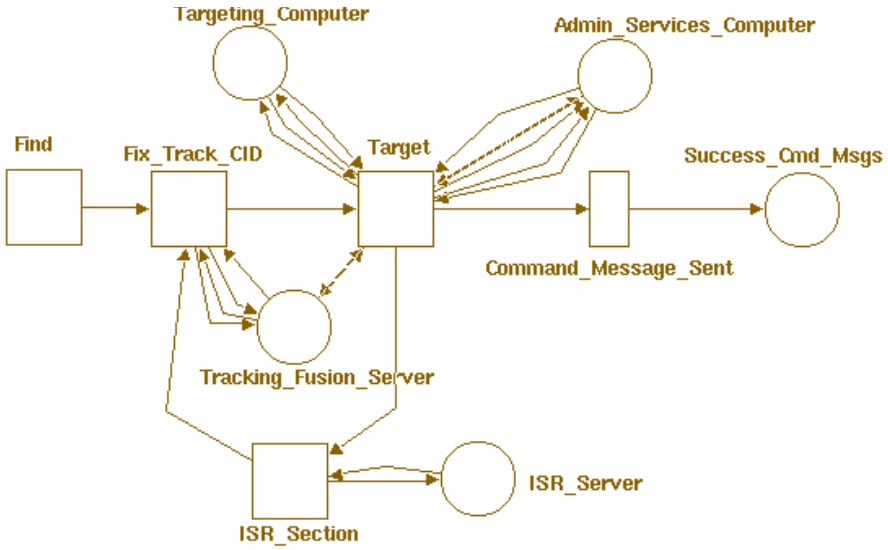

Figure 3: The TCT Cell workflow is modeled as a Colored Petri Net.

Target candidates enter the system with an exponentially-distributed inter-arrival time with mean one minute. Targeting candidates are identified in the Find subprocess. They are fixed, identified, and tracked by tasks in the Fix_Track_CID subprocess, possibly using information provided by the intelligence and image processing capabilities represented by the ISR_Section subprocess. The decision to prosecute a target and to assign either air or ground assets for the attack is made in the Target subprocess. For those targets prosecuted by an air asset, the final task performed is Command_Message_Sent, and the job reaches the successful completion state Success_Cmd_Msgs. If a target is prosecuted a surface asset, the Accept task transitions the job to the successful completion state Success_Task_Accepted (*not shown*). Surface asset prosecutions lessen the load on TCT Cell cyber-

resources, since air asset prosecutions require additional processing time by the Cell to review and approve the tasking order. A surface asset will use its own processing resources to determine whether it will accept a prosecution request.

Processing paths in the TCT workflow sometimes depend on time-dependent properties of a job. Figure 4 illustrates the different branches which a target candidate may follow depending on the results of the Attack Operations Decision Aid (AODA) Asset Selection task—if the target candidate is matched with a surface asset, it follows one branch, if an air asset, a different one, and if it is insufficiently valuable, it is discarded.

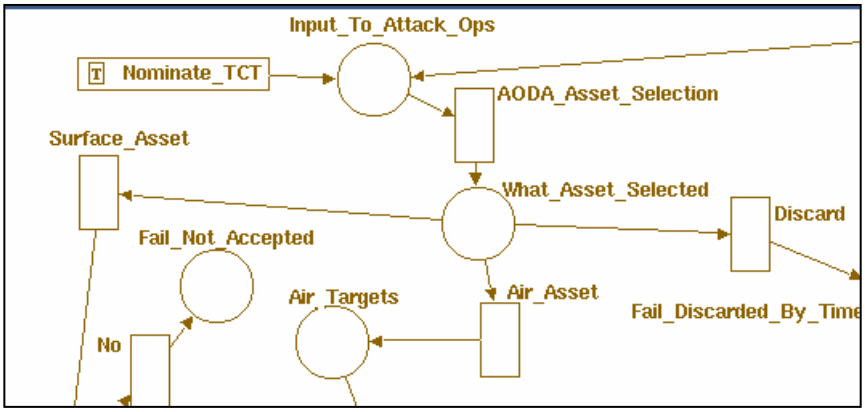

Figure 4. Target candidates may be prosecuted by surface or air assets depending on the result of the AODA Asset Selection task.

Petri Net models of the Find, Fix_Track_CID, ISR_Section, and Target subprocesses of the TCT model can be found in Appendix A of the EBCOTE-3 User's Manual.

SECTION 5

RESOURCE POLICY AND CONFIGURATION

In the EBCOTE model, cyber-attacks render cyber-resources unavailable for processing, or at least available at reduced capacities. Unavailability may be caused by sysadmins removing a compromised resource from the network to avoid further contamination or by other denial of service mechanisms. Cyber-CoAs are sysadmin responses to compensate for a cyber-attack by reallocating some processing tasks to other qualified resources (*activation*), or by changing an unqualified resource through hardware or software reconfiguration (*installation*). Thus, resource perturbations are the focus of our assessment of QoS impacts.

During this EBCOTE option, we enhanced our path reconstruction analysis to specifically incorporate flexible models of cyber-resources and resource configurations. Our resource models follow emerging business process and workflow management standards [20].

5.1 RESOURCE POLICY MODEL

In WfMS parlance, a **resource class** is a set of resource instances. Two types of propositional rules delimiting a resource class are distinguished—**roles**, which classify instances by functional similarity or qualifications, and **organizational units**, which classify instances by geography, command hierarchy, or business unit [1]. We focus only on roles in EBCOTE.

We model a role as a set of attributes possessed by a resource. This modeling is sensible because qualification of a resource for a task means that the resource possesses at least the attributes required to resolve a task's execution (e.g., evaluate a timing rule).

A **resource specification** is a set of resources which have the same attributes, and furthermore, those attributes have the same values. The attributes possessed by a resource must be consistent with the roles associated with its resource specification. For example, the role *secretary* is comprised of the attributes of Typing and Phone Skill. If a resource specification `junior assistant` has a role of *secretary*, then it must have values for Typing and Phone Skill. However, possessing a set of attributes associated with a role does not require that a resource specification has that role. For instance, a `senior analyst` may also have Typing and Phone Skill attributes, but not be designated with a role of *secretary*. A resource has a unique resource specification. Its attributes and values do not change during a workflow simulation unless an installation moves the resource from one specification to another.

In workflow resource management, three types of policies govern the allocation of resources to tasks—qualification, requirement, and substitution [13]. In EBCOTE-3, we focus on qualification policies, although our model may be extended to account for the more complex constraints imposed by requirement and substitution policies. A **qualification policy** associates resource classes with tasks through their roles. Our resource policy model allows a user to associate multiple roles with a task. Once

again, even if a specification has attributes necessary to resolve a task's execution, it may not be declared as qualified to perform it. For example, the `senior analyst` does not have the role of *secretary* and so could not be assigned to a task "Direct Incoming Call" in spite of having a Phone Skill attribute. A **requirement policy** specifies that a resource assignment to a task is valid only if attributes of the resource and attributes of the task satisfy some condition. For instance, a typist needs to be able to type a certain number of words per minute to be assigned the task of typing a 150-page long document. A **substitution policy** specifies rules for resources that may be adequately substituted if no resource meeting the preferred requirements is available.

We allow cyber-resources to change their resource specifications through *installation*. Installations are software and hardware changes that offer flexible response options to sysadmins. Installations are not cost-free, they require time to implement. Part of the resource policy model is a specification of the amount of time it takes to install a resource token from one resource specification to another.

Figure 5 shows the syntax for our resource policy model file. There are three sections. The first section declares all resource specifications, their attributes and attribute values, and roles. This declaration is followed by a set of qualification policies which identifies a role as appropriate for supporting a task. Not all tasks are supported by resources, but any resource specification which has a role qualified for a task must have the correct attributes to resolve the task. Particularly, this means that the resource spec must have any attributes the Perturbation Analyzer requires to evaluate the completion delay time distribution rule. The last section declares resource installation requirements in matrix form. Element *(i,j)* of the matrix records the time required to convert a resource with specification *i* into a resource with specification *j*, where a dash means that a conversion is not possible.

```
%resource specification declarations
<resource_spec_name_1>
      <attribute_1> = <real value>
      <attribute_2> = <real value>
      …
      <attribute_N> = <real value>
      roles = <role_1>, <role_2>, …, <role_N>
end
…

%qualification policies
qualify <role_1> for <task_name_1>
…
qualify <role_N> for <task_name_N>

install
%example matrix
-   - 1
0.5 - -
-   - -
```

Figure 5. Syntax for specifying a resource policy model.

Table 1 shows the five cyber-resource specifications in the TCT Cell model. A TCT Cell may have multiple instances of each available; for example, there may be multiple Targeting Computers. In

our scenario, server performance is characterized by the amount of RAM, and processing workstations by processor speed in GHz. The resources have roles that designate which members of the TCT Cell team may use them. The Emergency Server is not part of the nominal system configuration but may be brought on-line for ISR, Tracking, and Fusion tasks as needed.

TCT Cyber Resource	Attribute and Value	Role(s)
ISR Server	GB RAM = 0.5	ISR Technician
Tracking Fusion Server	GB RAM = 0.77	Target Intel Technician
Targeting Computer	GHz = 2	Targeting Officer
Admin Services Computer	GHz = 1.4	Targeting Officer, Team Chief
Emergency Server	GB RAM = 0.5	General Technician

Table 1. Cyber-resources in the TCT Cell example.

Table 2 shows which tasks each role is qualified to perform. The Final Approval Meeting and Team Chief Review Air Asset tasks are not required for surface asset prosecutions; they are required only when an air asset is assigned.

Role(s)	Tasks Qualified For
ISR Technician	ISR Processing
Target Intel Technician	Fix
	CID
	Process Candidate Target
Targeting Officer	Targ Officer Eval Target
	Final Approval Meeting
Targeting Officer, Team Chief	Team Chief Initial Review
	Team Chief Review Air Asset
	Final Approval Meeting
General Technician	ISR Processing
	Fix
	CID
	Process Candidate Target

Table 2. Qualification policies for TCT Cell tasks.

The Admin Services computers have the necessary software to help the Targeting Officer, but they are slower. Installations between Targeting and Admin Services computers are allowed, but time is needed to switch out processors (to convert an Admin computer into a Targeting Computer) or install software (to

allow a Targeting Computer to perform Admin tasks). Installations are also permitted among the three types of servers.

5.2 RESOURCE CONFIGURATIONS

Resource configurations describe the availability of cyber-resources to support a BMC3 workflow and their current usage. A configuration describes how many resources of each resource spec are at hand, which resource specs are activated to support which tasks, and any installation plans for modifying available resource tokens. The nominal configuration represents a pre-attack state, and rarely includes any plans for installation. Some resources, however, may not initially be assigned to support a task (e.g., due to routine maintenance).

Figure 6 shows the EBCOTE syntax for describing a resource configuration. The number of available tokens of each resource specification comes first. Then one or more resource specifications, all of which must have roles qualified to support the task in accord with the resource policy, are activated to support tasks. Finally, installation plans are described in matrix form—note that this installation-related matrix has different semantics from the matrix in the policy file. Here, element *(i,j)* of the matrix records the number of tokens with resource spec *i* that will be installed to resource spec *j*.

Multiple-step installations are not permitted in the EBCOTE model, i.e., we do not allow installations from resource spec *i* to *j*, and then on to *k*. Only the direct installation from *i* to *k* is considered as a CoA.

```
%total available resources in nominal configuration
<resource_spec_name_1>    <nonnegative integer>
<resource_spec_name_2>    <nonnegative integer>
…
<resource_spec_name_N>    <nonnegative integer>

task <task_name_1>
<resource_spec_name_1>
…
<resource_spec_name_M>

Installs:
%example install matrix
0 3 0
1 0 0
0 0 0
```

Figure 6. Syntax for a resource configuration.

Other configurations are perturbations to the workflow with respect to the nominal configuration, and have some impact on QoS. Thus configurations can dually represent both cyber-CoAs and effects of cyber-attacks. For instance, if a cyber-attack compromises the Targeting Computer in the TCT Cell, that token is no longer available (reflected in the first section of the configuration file). To complete any jobs, other workstations with adequate software must be tasked to perform functions normally handled by the

Targeting Computer. Such a cyber-CoA would be reflected in either the second section (an activation CoA) or the third section (an installation CoA).

An example post-attack configuration is shown in Figure 7. There are normally three Targeting Computer workstations online, but a cyber-attack has caused the sysadmins to remove them from the network to prevent further contamination. Under these circumstances, the Targ Officer Eval Target and Final Approval Meeting tasks are unsupported—no TCTs can be fully processed in this configuration. A couple of CoAs consistent with the resource model are activating other computers (such as the Admin Services) to do these unsupported tasks or performing an installation on an Admin Services computer to enhance its performance. The Perturbation Analyzer (Section 6) can evaluate the QoS impacts of these CoAs, and the cyber-CoA Search algorithm (Section 7) can search for these and better alternatives.

```
ISR_Server 5
Tracking_Fusion_Server 10
Targeting_Computer 0
Admin_Services_Computer 2
Emergency_Server 0

task ISR_Processing
ISR_Server

task Fix
Tracking_Fusion_Server

task CID
Tracking_Fusion_Server

task Process_Candidate_Target
Tracking_Fusion_Server

task Targ_Officer_Eval_Target
Targeting_Computer

task Final_Approval_Meeting_1
Targeting_Computer

task Team_Chief_Initial_Review
Admin_Services_Computer

task Team_Chief_Review_Air_Asset
Admin_Services_Computer

task Final_Approval_Meeting_2
Admin_Services_Computer

Installs:

0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
0 0 0 0 0
```

Figure 7. A post-attack configuration that has taken Targeting Computers off-line.

5.3 RESOURCE CONFIGURATION CONSTRAINTS

Resource configurations form the space of solutions for the QoS optimization problem of Section 7. A resource configuration, relative to the nominal configuration, is then represented by the following sets of decision variables:

$\mathrm{A} = \left[\alpha_{ij} \right]$ is a matrix of variables such that $\alpha_{ij} = 1$ if the resource tokens belonging to resource specification i have been allocated to support workflow task j and $\alpha_{ij} = 0$ otherwise.

$\mathrm{T} = \left[\tau_{ij} \right]$ is a matrix of variables such that τ_{ij} is the number of resource tokens initially belonging to resource specification i that have been transferred, by installation, to resource specification j.

Any resource configuration change must be *feasible* given the starting configuration, thus these variables must obey the following constraints:

Constraint I : Let $\mathbf{I} = \left[I_{ij} \right]$ be the installation matrix from the resource policy model. By definition, I_{ij} is the time necessary to convert resource specification i to j by installation. Let $\tilde{\mathbf{I}} = \left[\tilde{I}_{ij} \right]$ be the matrix with an entry of 1 where the corresponding installation time is finite, and 0 otherwise. Then,

If $\tilde{I}_{ij} = 0$, then $\tau_{ij} = 0$. (We cannot convert resource specification *i* into *j*.)

Constraint II : Let N_i be the number of resource tokens belonging to resource specification i in the perturbed configuration. Then

For each j, $\sum_j \tau_{ij} \le N_i$. (We cannot convert by installation more tokens than initially available.)

Constraint III : Let $\mathbf{R} = \left[r_{ij} \right]$ be such that $r_{ij} = 1$ if resource specification i has a role qualified to support workflow task j; $r_{ij} = 0$ otherwise. Then

$\alpha_{ij} \le r_{ij}$. (Resources can be assigned to task only if role is qualified.)

Our cyber-CoA search algorithm generates candidate CoAs in terms of the decision variables A and T. Constraint I and Constraint III are easy to enforce—these constraints require that any solution is found in the subspace where the respective variables must be zero. By identifying variables whose values must be zero before running the optimization and removing them from the optimization solution vector, we reduce our search space and improve performance.

SECTION 6

PERTURBATION ANALYZER ENHANCEMENTS

The Perturbation Analyzer (PA) component implements a path reconstruction analysis (PRA) algorithm for propagating changes to a stochastic process without full discrete-event simulation of the changed process. Given a sample history of processing events for the nominal workflow, PA estimates the QoS impact of a resource configuration change. In EBCOTE-2, we demonstrated the computational benefits of using PRA compared with full simulation, which we review in Section 6.1. By propagating the limited effects of a perturbation using a previously generated history, we observed a factor of 6 to 7 reduction in the time required to evaluate a configuration, while accurately estimating QoS within a few percent of that predicted by simulation.

In EBCOTE-2, PA handled two types of perturbations. First, PA predicted the impact of slight modifications to routing decision rules based on timing deadlines. An example of this type of decision rule is routing a job along different branches of a process according to the time it has been in the system—if the breakpoint for making the decision is modified, how does this affect QoS? This type of perturbation was not the main focus of EBCOTE-3. The second perturbation handled in EBCOTE-2 was a modification to parameters in a task's completion time distribution. For instance, if TCT weapons selection takes one minute longer on average, how does this affect QoS?

This second type of perturbation corresponds to the effect of resource reconfigurations in EBCOTE-3. EBCOTE-2, however, did not utilize a resource policy model or track resource usage during perturbation analysis. Several enhancements to the path reconstruction algorithm were needed, including the incorporation of resource usage audit data in the workflow history log and accounting for installation delays. We also improved the algorithm to account for jobs in progress when the cyber-attack occurs and to evaluate QoS using user-specified criteria for job priority and deadlines.

We review the PRA algorithm implemented in EBCOTE-2 and our results in Section 6.1, then discuss the essential enhancements required and implemented in EBCOTE-3 in the remainder of the Section 6.

6.1 EBCOTE-2 BASIC PATH RECONSTRUCTION ALGORITHM

Discrete-event simulation is not an altogether efficient way to analyze perturbations in a workflow model. A perturbation in a large workflow will rarely affect all activities therein, obviating the need to simulate the entire workflow. Rather, a perturbation will typically affect a narrow thread of activities. This is normally true even if the perturbation affects many different segments of the workflow. A typical perturbation is akin to a web: it may span a wide area, but it does not blanket all of the area that it spans. Various perturbation analysis methods have been developed to exploit these observations [14]. The problem, is that we cannot anticipate the span and coverage of this 'web' beforehand and direct the

simulation to just those threads involved. Instead, discrete-event simulation must blanket the entire workflow to discover which portions are actually affected by a perturbation.

The ideal perturbation analysis method would discover and focus its attention on only those activities that are affected by the perturbation. This, in a nutshell, is the motivation for Path Reconstruction Analysis (PRA) developed by Cassandras [8]. The idea is to take task processing histories that had been recorded under nominal conditions, estimate which ones *would* have been affected by the perturbation, and construct new processing histories as if the perturbation had been in effect.

We have implemented an enhanced version of PRA to analyze disruptions to BMC3 workflow. We introduce the following terms to describe it:

1. Event: the transformation of a task from one completion state to another
2. Event Type: a *type* of task going from one completion state to another *completion* state. These are determined by the business rules for the workflow system under study.
3. Enabled Event: an Event for a specific task that can occur (i.e., its pre-conditions have all been met), and with a proposed duration, but without a scheduled start time
4. Scheduled Event: an Event for a specific task that has a specified start but an unknown completion time
5. Completed Event: an Event for a specific task with known start and completion times
6. Sample Path: the history of a specific task as it passed through (or is passing through) the workflow system; it is a sequence of Completed Events.

6.1.1 Path Reconstruction Algorithm Pseudocode

Initialization

1. Record the completion state of tasks the workflow at the current time
2. For each task currently in the workflow:
   - Locate a task from the workflow logs that reached a comparable completion state, and retrieve its Sample Path
   - Find the Completed Event from that Sample Path that brought the task to this Completion State, and store in a list of Completed Events (CompletedEventList)
   - For each Completed Event that appears thereafter in the Sample Path:
     - Create a corresponding Enabled Event
     - Set the proposed duration for this Enabled Event to the duration of the Completed Event
     - Store the Enabled Event in a list of Enabled Event (EnabledEventList)
3. Locate and remove the Completed Event from CompletedEventList that had the latest completion time. Create a corresponding Scheduled Event , give it the start time for the Completed Event, and store it in a list of Scheduled Events (ScheduledEventList); this list will be sorted by start time.

At the completion of the Initialization step we have a single event in ScheduledEventList, an initial set of completed events in CompletedEventList, and a large number of prospective Events in EnabledEventList.

Path Reconstruction

Repeat

1. Remove the event at the head of ScheduledEventList, determine its completion time, and add it to CompletedEventList. If the Event is not directly affected by the disturbance, then the completion time is simply the start time + the nominal duration. Otherwise, use a model of the disturbance to compute the duration in order to establish the completion time
2. *Locate* events in the EnabledEventsList that are now enabled by the occurrence of this event, *schedule* their completion times, and place them in ScheduledEventlist

Until ScheduledEventList is empty

We have necessarily omitted several details in outlining the method, and we fill them in now.

Step 2 of the Initialization phase sets up prospective event histories for tasks that are currently in the workflow system. We can extend this to include new tasks forecasted to enter the system: they will simply enter with a nominal initial completion state, and the initialization actions in Step 2 will take care of the rest.

Perturbations may alter processing paths within the workflow system, and send tasks down different processing paths than the nominal event history would have suggested. We can handle this condition with the following additional actions in Step 1 of the Path Reconstruction phase:

- Eliminate the nominal events from EnabledEventsList corresponding to the task in question
- Locate a task from the nominal workflow log that went down the branch and retrieve its event history. Create Enabled Events corresponding to those events (following the rules used during Initialization), store them in EnabledEventList., and continue on to Step 2

We have also glossed over how we handle ‘time,’ and whether we treat time in a relative sense or in an absolute sense. The simple answer is that we do both, depending on the purpose in mind. Workflow logs typically record event start and completion times against an absolute reference, but the nominal duration (a relative measure) is the essential information we need to obtain from the logs during the Initialization phase. During the Path Reconstruction phase, we first set time to the current system time and then derive event start and completion times offset from that time during the path reconstruction analysis.

We have also glossed over how we handle resource contention. Our PRA implementation treats the release of a resource from an activity as a special ‘resource release’ event. Step 2 of the Path Reconstruction phase normally locates *all* events in EnabledEventsList that become enabled when an event occurs. When a resource release event occurs, however, we locate a single event in EnabledEventsList; the selection of a single event accounts for resource contention.

6.1.2 Illustrative Analysis for a DoS Cyber-Attack

We illustrate the efficacy of our PRA implementations with a sample case study involving a denial of service (DoS) attack on our TCT Cell BMC3 example. Figure 8 depicts cyber-assets and their functions in the TCT Cell.

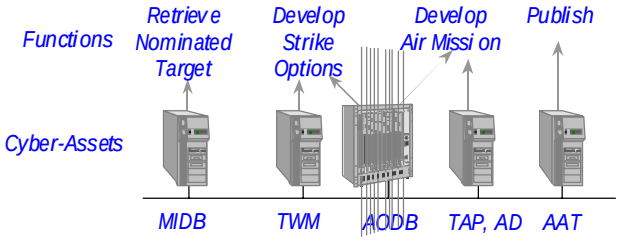

Figure 8. The Key Cyber-Assets and Functions for our Hypothetical Timce Critical Targeting System

The "tasks" in this system are requests to strike TCTs, and they pass through four key functions:

- The system validates each request to confirm that the track qualifies as a TCT. The system retrieves key information about the track and about neighboring enemy units (e.g., air defense units) from intelligence databases (e.g., the MIDB). In the simplified workflow and resource model described in Section 4.2 and Section 5.1, these functions are served by the ISR and Tracking/Fusion Servers.
- The system develops strike options against validated requests by developing new air missions and by locating existing air missions that can be diverted to strike the TCT. The system retrieves existing air missions and mission planning data from the Air Operations Data Base (AODB) and matches strike aircraft to strike requests using the Target Weaponeering Module (TWM). In the simplified workflow and resource model described in Section 4.2 and Section 5.1, these functions are served by the Targeting Computer.
- The system develops new air mission packages and establishes specific instructions for the strike aircraft and for supporting assets (e.g., air controllers, electronic countermeasure support, surveillance support, etc). The system both retrieves information from the AODB and updates the missions therein. The system prepares the missions instructions using the Theater Air Planning (TAP) system. In the simplified workflow and resource model described in Section 4.2 and Section 5.1, these functions are served by the Admin Services Computer.
- Finally, the system publishes and disseminates the new air mission orders.

The workflow must be able to complete these four functions within a specific amount of time after a request appears, and we set that deadline to 5 minutes in our example. We hasten to add that people are instrumental players in the workflow, though they are not depicted in Figure 2, and they decide whether tasks should be sent on to downstream functions, re-worked by upstream functions, or abandoned outright.

Figure 9 illustrates the time spent in the workflow system for a stream of TCT requests under nominal conditions. The X-axis depicts the time of arrival for a request and the Y-axis depicts the time required to service that request. The average service time is roughly 2.5 minutes, and the maximum time was less than the 5 minute deadline.

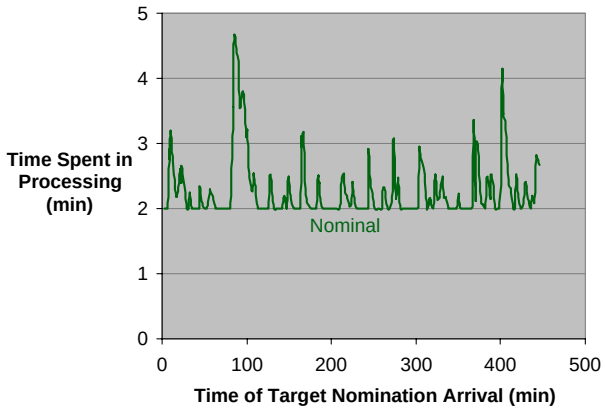

Figure 9. Service Times Under Nominal Conditions.

We consider a DoS attack on the AODB. It is a key resource in this workflow, supporting the two functions listed above along with a host of low-level functions distributed throughout the processing chain. A disruption in this application, therefore, will reach out into many activities in the workflow.

The DoS attack increases the number of database accesses seen normally by a factor of four. We consider two possible ways variations of this attack. The first variation simply increases database processing time by that factor. In the second variation the malicious access requests compete with legitimate requests, possibly blocking connections to the AODB.

We analyzed both DoS variations using PRA and using Monte-Carlo Simulation using ALPHATECH's Colored Petri Net simulator, ALPHA/Sim [3]. We use the same arrival profile of TCT requests used to generate the nominal profile illustrated in Figure 3. The PRA uses the workflow log recorded for the nominal condition, and simply introduces perturbations in the processing times for those activities that use the AODB.

Figure 10 depicts the history of service times for the first DoS variant. The Monte-Carlo simulation results appear in the top-most curve, and the PRA results appear immediately below. The two perturbation analyses agree almost perfectly. The average service time increases to roughly 3.1-3.2 minutes, but all service times stay within the 5 minute deadline.

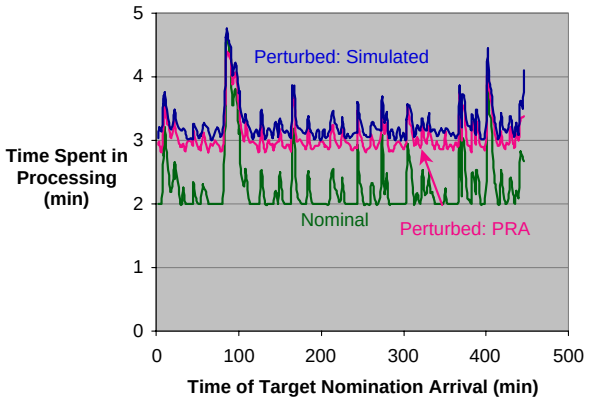

Figure 10. Service Times When the DOS Attack Simply Retards Processing

Figure 11 depicts the history of service times for the second DOS variant. As before, the Monte-Carlo simulation results appear in the top-most curve, and the PRA results appear immediately below. This attack creates a bottleneck at the AODB, and service times grow as new TCT requests arrive at a congested workflow. The two perturbation analyses do not agree perfectly, but PRA nonetheless predicts both the impact and its dynamics quite well. This is all the more remarkable when we recall that these predictions are derived using the nominal workflow event history!

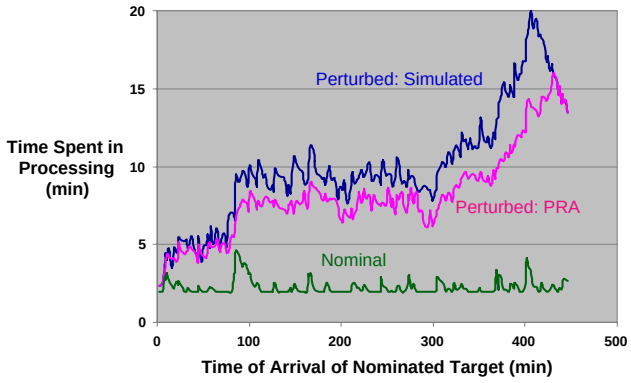

Figure 11. Service Times When the DOS Attack Blocks Database Access Requests

PRA is significantly faster than discrete-event simulation, as illustrated in Figure 12. Here we report runtimes for both vs. the size of the workflow system (here, measured as the number of tasks in the workflow). PRA is a factor of 6 to 7 times faster than discrete event simulation.

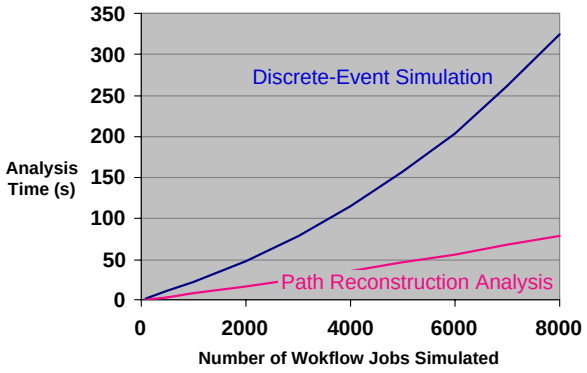

Figure 12. Runtime vs. Workflow Size for Monte-Carlo Simulation and PRA

6.2 EBCOTE-3 PRA (WITH ENHANCED RESOURCE MODEL)

The key modification to the algorithm for EBCOTE-3 is easily summarized: task completion delay distributions are now functions of resource attributes. Perturbations to delay distribution parameters are then determined by differences between the attributes of resources supporting a task in the nominal and perturbed cases.

Table 3 shows a sample from a TCT Cell job history log. The log includes the Job ID, Job Class, the task associated with the event, completion delay and completion time, the state of the job after the event, and a relevant attribute from the supporting resource. In this example, resource-supported tasks use the Tracking Fusion Server, so the important value is the RAM of the server. We will perturb this sample history by substituting two Emergency Server tokens (RAM = 0.5) for the Tracking Fusion Server tokens on the Fix task.

Job	Cls	Transition	Delay	Complete	Next State	RAttr
1	1	Record_Detect_Time	0.000000	2.226477	Target_To_Fix	N/A
2	1	Record_Detect_Time	0.000000	2.143620	Target_To_Fix	N/A
3	2	Record_Detect_Time	0.000000	2.269640	Target_To_Fix	N/A
2	1	Fix	0.470978	2.614600	Is_Target_Fixed	0.77
2	1	Determine_Track_Time	0.000000	2.614600	Target_To_Track	N/A
1	1	Fix	0.734323	2.960800	Is_Target_Fixed	0.77
1	1	Not_Fixed	0.000000	2.760800	Fail_Not_Fixed	N/A
2	1	CID	0.576590	3.191190	CID_Done	0.77

Table 3. Sample data from a TCT Cell job history log.

Using the PA algorithm, we begin with the first three events as SCHEDULED. We derive the event sequence from the workflow model, thus events corresponding to the Fix task for jobs 1 and 2 are ENABLED.

The first event in the ENABLED list is job 2's Fix activity. An Emergency Server resource token is available. The nominal delay for the Fix activity in the log is 0.47, but that was when the supporting resource had 0.77 GB of RAM. To calculate the perturbed delay, PA consults the Fix task timing rule in

the workflow model—the distribution is triangular with a minimum of 0.3, a mode of 0.5, and a maximum that depends on the inverse of the RAM value.

If Z is the nominal delay, then an estimate of the perturbed delay is given by the formula

$$Z' = F^{-1}(F(Z; 0.3, 1/(RAM), 0.5); 0.3, (1/RAM), 0.5),$$ where F = Triangular CDF

In this example, Z' = 0.52441; this is the new delay. We update the perturbed log to reflect this event as SCHEDULED at t = 2.143620 + 0.52441 = 2.668061. An Emergency Server token's next availability time is updated to 2.668061.

The downstream queue for Fix is the Is_Target_Fixed state. This state has an associated routing decision rule that governs the next task in the process. When this rule is evaluated under the perturbation, Determine Track Time is no longer the next activity event to be enabled for job 2. Instead, a Not_Fixed activity event will be enabled.

The Path_Update procedure is now used to supply a fictitious history for this new branch. Another job in the log does go down this path—job 1 has the same job class as job 2 and followed the Not_Fixed branch. The Not_Fixed activity event for job 1 is copied, and assigned as the child event for job 2's Fix event. All children along that path for job 1 are also copied, and the job IDs are updated to associate the copies with job 2.

The algorithm proceeds by considering the next ENABLED event in the list, the Fix activity for job 1.

6.3 INSTALL DELAYS

Although assignment of resource classes to support tasks (activation) is immediate in our model, the process of transferring tokens from one class to another (installation) may involve a time delay. To model this cost of installation reconfigurations, we augmented the PA to intelligently add and remove tokens from a resource spec. The following example illustrates our modifications:

Suppose Targeting Computer contains five tokens, with availability {4, 8, 10, 11, 13} and Admin Services Computer has four tokens with availability {5, 8, 9, 15}. Targeting Computer supports the Eval Target task, and for our example, assume that Eval Target always has a delay of 3. Consider the situation where one token will be transferred from Targeting to Admin; according to the installation matrix, the delay for installation between them is 6 time units.

Although installation begins immediately at t = 0, our model assumes that the token undergoing installation does not have to remain idle, and is still available to initiate an Eval Target activity up until t = 6. In fact, we also assume that even once installation is completed, any activities using the installed token are allowed to finish before the token is removed from Targeting and made available to Admin.

We mark the Targeting Computer token with availability at t=5 as the one for install.

- Job A arrives at Eval Target, the activity is ENABLED at t = 4. It uses a resource token available at t = 4, and the activity is SCHEDULED for completion at t = 7, leaving the Targeting Computer spec with availability times {7, 8, 10, 11, 13}.

- Job B arrives at Eval Target, and the activity is ENABLED at t = 5. Under the old PA, it would have used the token available at t = 7, however, now the installation delay has elapsed at t = 6. The token originally flagged for install is deleted from Targeting (now availability = {8, 10, 11, 13}) and added to Admin with the same availability time (making Admin availability = {5, 7, 8, 9, 15}).
- Job B's Eval Target is SCHEDULED, then, at t = 8, and leaves the Targeting Computer spec with availability times {10, 11, 11, 13}.

This model *does possibly penalize some jobs supported by the target resource specification (Admin Services).* For instance, if two jobs had arrived at a task supported by Admin Services Computer before t = 7, the second job would have committed to beginning that task at t = 8 due to Admin Services availability. The second job is committed to a resource token with no foreknowledge that an install is destined to make one available earlier at t = 7. For evaluating QoS, this penalty is more than offset by our assumption allowing the installed resource to be fully used during its installation.

6.4 IN-PROGRESS JOBS

The PA evaluates the impacts of a resource configuration change on QoS by performing path reconstruction on a job history log that is representative of current and anticipated workload. If the number of jobs currently in the system, however, is an non-negligible fraction of the total number of jobs in the sample history, a projection of cyber-CoA effects on these jobs in-progress is necessary for an accurate assessment of QoS.

EBCOTE-3 uses several pieces of information, obtainable from WfMC-compliant WfMS audit data, about in-progress jobs:

- Job ID
- Job class
- Job arrival time
- The next task ID, if in queue, or the current task ID if an activity is in-progress for that job. We assume that routing decision rules have already been evaluated for queued in-progress jobs at the time PA begins, so that the transition the job is waiting for is well-defined.
- If an activity is in-progress for the job, the time the activity was started (in PA terms, the time after the corresponding event is marked ENABLED when a resource is available).
- If an activity is in-progress for the job, the resource spec of the token supporting the activity.

The PA cross-checks that the resource spec of the token supporting the activity is

1. qualified to support the activity according to the resource model.
2. assigned to support the activity by the current (nominal) configuration.
3. contains a non-zero number of tokens in the nominal configuration.

We use the following procedure to integrate in-progress jobs into the job history log:

- If the job is in queue, we simply find a candidate event (with the same job class) for the awaited task and copy the path, just as we do with decision-rule branching (Path_Update, see Section 6.1). We ENABLE that event at t = 0.
- If the job is currently being processed in an activity, we need to:
    1. Find a candidate event for the task corresponding to the in-progress activity, with the same job class, and copy the path.
    2. ENABLE the event at the activity start time.
    3. Calculate the delay for the activity using the copied path's delay (taking into account the possible perturbation resulting from differences in the attributes of token supporting the copied activity from the attributes of the token supporting the actual in-progress activity).
    4. SCHEDULED time = max (0, ENABLED time + delay). Thus, if the calculated delay isn't enough to keep the activity in-progress up to the present time, assume the delay will be longer and that the activity will finish right when PA begins.
    5. Set the availability time of one of the tokens in the supporting resource spec to be max (0, ENABLE time + delay).

6.5 USER/TASK QOS PROFILE

After the PA generates a perturbed history, it calculates the QoS metric according to the input provided in the User/Task QoS Profile. This profile assigns to each job class a deadline D (relative to the job's in-system arrival time) and priority weight P. Suppose that a job arrives in the system at time t_a and enters a success state at time t_s.

- If $t_s =< t_a + D$, then P is added to the QoS score.
- Otherwise, $P * \exp(D-(t_s - t_a))$ is added to the QoS score. This allows jobs that succeed, but do not meet their deadline, to count for an increasingly smaller amount as time in system elapses.
- Jobs which reach a failure state add nothing to the QoS score.

SECTION 7

CYBER-COURSE OF ACTION SEARCH

From the discussion in Sections 5.3 and 6.3, we can formulate the BMC3 QoS optimization problem as follows. Let *Succ* be the set of jobs reaching a completion state, $t_a^{(i)}$ the time of arrival of job $i \in Succ$, $t_s^{(i)}$ the time job *i* reaches its completion state. *Succ*, $t_a^{(i)}$, and $t_s^{(i)}$ are all complex stochastic functions of the resource configuration variables A and T of Section 5.3. (We vectorize the matrices A and T to ease the expression of the optimization problem.) Thus our optimization problem can be written:

$$\min_{\mathrm{A,T}} QoS = \sum_{i \in Succ(\mathrm{A,T})} V(t_a^{(i)}(\mathrm{A,T}), t_s^{(i)}(\mathrm{A,T}))$$

where

$$V(t_a^{(i)}(\mathrm{A,T}), t_s^{(i)}(\mathrm{A,T})) = \begin{cases} P \text{ if } t_s^{(i)} \le t_a^{(i)} + D \\ Pe^{(D-(t_s^{(i)}-t_a^{(i)}))} \text{ otherwise} \end{cases}$$

subject to the three constraints of Section 5.3.

The complexity of the QoS objective function requires the use of an oracle to evaluate it. One oracle is discrete event simulation over the Petri Net workflow model. We use the faster PRA algorithm (Section 6) in EBCOTE.

7.1 SEARCH ALGORITHM INTUITION

Our approach to search in the BMC3 QoS optimization problem is motivated by the heuristic use of the *approximate co-state* to suggest improving resource configurations. The approximate co-state is a change in a job's expected value from a given state given a resource configuration change. This concept is discussed in [4], where a formal expression for off-line calculation of approximate co-state is given. For on-line CoA search, however, we need a fast way to estimate these values.

The use of approximate co-state to generate a search direction is equivalent to a hill-climbing algorithm using an estimated gradient. Given the complexity of our objective function, gradient estimation must occur using PRA or some other oracle to obtain evaluations of nearby configurations.

In EBCOTE-3, we developed a surrogate optimization algorithm, following Gokbayrak and Cassandras [12], that leverages path reconstruction. Intuitively, surrogate optimization uses a steepest ascent search on a continuous relaxation of the discrete stochastic problem, then projects the optimal continuous problem's solution onto the nearest feasible discrete problem solution. The oracle, however,

cannot meaningfully evaluate QoS for non-integer values of the activation and installation variables, hence another means of estimating the gradient is necessary.

Gokbayrak and Cassandras show that the gradient at a continuous solution may be estimated by evaluating the oracle at a small number of nearby discrete solution vectors called a *selection set*. If *N* is the number of dimensions in the optimization problem, then we need only evaluate *N+1* discrete solutions to estimate the gradient. Given the need to repeatedly evaluate alternative configurations, the fast PRA oracle is critical for providing real-time performance. *Since PRA is 6-7 times faster than discrete event simulation, our surrogate optimization algorithm is 7N times faster than an equivalent algorithm using simulation, clearly two or more orders of magnitude improvement on most problems.*

In Section 7.2, we describe the surrogate optimization algorithm as applied to EBCOTE, and provide a small example. Our empirical evaluation of the algorithm's performance follows in Section 7.3, and we conclude with a summary of known issues with the algorithm and lessons learned.

7.2 COA SEARCH ALGORITHM

7.2.1 Main Loop

7.2.1.1 SOLUTION VECTOR

We separate our optimization problem into two subproblems for the two sets of COA variables, the activation variables denoting whether resource class *i* is assigned to support task *j*, and the installation variables denoting how many resource tokens of class *i* are transferred to resource class *j*. This separation is motivated by the very different characters of the two classes of variables. The activation variables all have domain {0,1}. If an installation is allowed at all, an installation variable can take on any nonnegative integer as a value, subject to the constraint that the sum of installations from any resource spec cannot exceed the total number of tokens initially available in that specification.

If a resource class does not have a role qualified to support a task, then the corresponding activation variable must be zero. For the activation subproblem, we project our solution vector onto the subspace of feasible activations and eliminate those variables from search.

Likewise, the installation matrix in the resource policy file may indicate that a token resource class *i* cannot be installed to become a token in resource class *j*. Those installation variables must be zero. In the installation subproblem, we project our solution vector onto the subspace of where installations may feasibly be nonzero and eliminate those variables.

7.2.1.2 SUBPROBLEM ALTERNATION

We search for CoAs in the subproblems iteratively, holding fixed any previous solution to the other subproblem (with an exception discussed in Section 7.2.1.3), and halting when no improvement is found in the subproblem from the previous solution.

We initially perform steepest ascent search on both subproblems and take the better solution produced between the two, and then alternate subproblems thereafter. For example, the COA Search algorithm would

- Solve the activation subproblem assuming initial installation (e.g., QoS = 45 after steepest ascent)
- Solve the installation subproblem assuming initial activation (e.g., QoS = 55 after steepest ascent)
- Select the configuration with the better answer. (= Installation 1)
- Solve activation assuming Installation 1. (= Activation 2)
- Solve installation assuming Activation 2. (= Installation 2)
- Solve activation assuming Installation 2. (= Activation 3)…and so on until a subproblem search does not find an improving configuration.

Trying both subproblems first was motivated by our initial experiments with the algorithm. We found, for instance, that (Activation 0, Installation 1) may be a better solution than (Activation 1, Installation 0), yet this may not be reachable from the latter if the solution (Activation 1, Installation 1) is not itself an improvement over it. The search was biased to finding the local minima resulting from resource activation CoAs. This issue disappears after the initial subproblem gets solved, and we are free to strictly alternate subproblems thereafter.

7.2.1.3 Assumed Activation for the Installation Subproblem

The installation subproblem assumes the activation solution found in the previous subproblem iteration with one important exception. For resource specs containing no tokens, these specs should be activated for all tasks for which they are qualified. Note that this modified activation does not affect the QoS of the initial configuration.

Without this exception, we found that CoA search could miss this two-step CoA improvement: 1) Install tokens to classes not currently activated, then 2) Activate them in the subsequent activation subproblem iteration. CoA search misses this solution because neither step is separately an improvement in QoS.

7.2.2 Surrogate Optimization

The surrogate optimization loop attempts to optimize QoS over the variables corresponding to either the activation or installation subproblem while leaving the remaining variables fixed for purposes of evaluating test configuration QoS. We will refer to the vector of variables subject to optimization for either subproblem using ρ.

On the first pass at the activation and installation subproblems, ρ_0 is the configuration resulting from the cyber-attack before reconfiguration occurs. On subsequent passes ρ_0 reflects the solution returned from the last subproblem. ρ_0 is guaranteed to be a feasible solution to the continuous relaxation problem.

The procedure **`determine_approx_discrete_set`** finds a set of discrete points near continuous solution (Section 7.2.4.1).

The **`perturbation_analyzer`** oracle finds a QoS evaluation for each discrete point in set (Section 6).

We then **compute_gradient** at the current continuous problem solution using the perturbation analyzer's QoS evaluations at the discrete points (Section 7.2.4.2).

The continuous solution is updated using steepest ascent. The step-size schedule η_n is one of the areas where the user can tune the algorithmic performance. For EBCOTE $\eta_n = \frac{K}{n+1}$, where *K* is a user-defined parameter, and *n* is the iteration number. This schedule takes smaller steps as we presumably get nearer to a local maximum.

```
surrogate_optimization( ρ0 ) :
ρ = ρ0 ;
do
{     n = 0;
       S(ρ) = determine_approx_discrete_set( ρ );
      for r^i ∈ S(ρ)
            Qd(r^i) = perturbation_analyzer( r^i );
      end
      ∇Qc(ρ) = compute_gradient({ Qd(r^i) });
      ρ' = ρ + ηn ∇Qc(ρ) ;
      ρ = enforce_continuous_feasibility( ρ' );
} while Stopping_Condition;
return enforce_integral_feasibility( ρ );
```

Figure 13. Pseudocode for Surrogate Optimization.

The **enforce_continuous_feasibility** (Section 7.2.4.3) ensures that the updated continuous solution obeys the relaxed constraints for the continuous problem by projecting the current solution onto the nearest feasible one.

We continue with steepest ascents until a **Stopping_Condition** is reached. The choice of stopping condition is one of the areas where we can tune the algorithmic performance. A standard stopping condition is to terminate steepest ascent when the gradient becomes sufficiently small. We describe the actual set of stopping conditions used in EBCOTE in Section 7.2.3.

The outcome of steepest ascents is a locally optimal solution to the relaxed continuous problem. We then project the feasible continuous solution onto a feasible solution for the discrete problem using **enforce_integral_feasibility** (Section 7.2.4.4).

7.2.3 Stopping conditions

1) Activation and Installation subproblem stopping condition 1.

Hill-climbing for both activation and installation subproblems stops when the gradient becomes small.

$$\text{Stopping condition 1}: \left\|\nabla Q_c(\rho)\right\| < \varepsilon_1$$

2) Activation and Installation subproblem stopping condition 2.

Hill-climbing for both activations and installation subproblems stops when a cycle is detected; that is, if hill-climbing revisits the neighborhood of a previously revisited solution to the continuous problem.

$$\text{Stopping condition 2}: \exists \rho_k \mid \left\|\rho - \rho_k\right\| < \varepsilon_1$$

3) Installation subproblem stopping condition 3.

In each loop of the steepest ascent solution to the continuous surrogate problem, there are three continuously approximated CoAs considered:

A) The solution generated in the previous iteration, $\mathbf{x}$.

B) The incrementally updated solution obtained by adding step size times gradient, $\mathbf{x}'$. This solution may lie outside the feasible region.

C) The projection of the incrementally updated solution onto the feasible region, i.e., by enforcing the continuous constraints, $\mathbf{x}''$.

This stopping condition for installation compares the projected point to the previous solution. If these points are close together, yet our increment was large, then the gradient's improvement was largely in an infeasible direction. This suggests an additional stopping condition for installation.

$$\text{Stopping condition 3}: \frac{\left\|\mathbf{x}'' - \mathbf{x}\right\|}{\left\|\mathbf{x}' - \mathbf{x}\right\|} < \varepsilon_2$$

7.2.4 Other algorithmic routines

7.2.4.1 DETERMINE_APPROX_DISCRETE_SET

This routine determines a set of $|\rho|+1$ discrete points nearby a continuous solution ρ that can be used to evaluate a gradient for the continuous problem (through path reconstruction simulation). Note that the elements of this set need not be feasible solutions to the discrete optimization problem (they may lie outside the constraints). Thus, we determined a convention for evaluating infeasible discrete solutions in the perturbation analyzer:

- For the activation subproblem, an activation variable value of 2 is equivalent to a value of 1; it indicates that the associated resource spec is activated for the associated task.
- For the installation subproblem the number of tokens installed *from* a resource spec might (infeasibly) exceed the number available in the nominal configuration. If this occurs, the PA marks as many tokens for installation as it can. It then keeps track of the deficit, and when an install occurs that requires more tokens than are in the source resource spec, it creates additional tokens in the target resource spec with availability time equal to their hypothetical install time.

The PA is then allowed to evaluate hypothetical, but infeasible configurations for the purpose of gradient estimation only. Reported CoAs are always feasible.

Gokbayrak and Cassandras demonstrate that the convex nature of our constraints for this problem guarantees that at least one of the selection set points is feasible.

determine_approx_discrete_set(ρ) :

$\tilde{\rho} = \rho - \lfloor \rho \rfloor$;

I = {1..length($\tilde{\rho}$)};

$\mathbf{v} = \tilde{\rho}$;

while $I \neq \varnothing$

{ $\tilde{r}^i = \sum_{j \in I} e_j$, where $i = \arg\min\{v_j, j \in I\}$, and e_j is the unit jth component vector;

$\alpha_i = v_i$;

$\mathbf{v} = \mathbf{v} - \alpha_i \tilde{r}^i$;

$I = I \setminus \{i\}$;

}

$\tilde{r}^0 = 0$;

$\alpha_0 = 1 - \sum_{i=1}^{N} \alpha_i$;

return $S(\rho) = \{r^i \mid r^i = \tilde{r}^i + \lfloor \rho \rfloor \text{ for } i = 0,..,N\}$;

Figure 14. Psuedocode for determining the "selection set" of discrete points used to approximate the gradient at a solution to the continuous surrogate problem.

The selection set for a solution to the continuous problem is found by considering the non-integer part of the solution, i.e. the vector of fractional parts of each component. We calculate a set of discrete approximating points around this non-integer part, which have components of either 0 or 1. The non-integer part is a convex combination of these approximating points. The set of approximating points has $|\rho| + 1$ members, and the last is always a vector of zeros.

The selection set itself is obtained by adding the integer part of the continuous solution back to each member of this set of discrete approximating points. Figure 15 illustrates the results schematically.

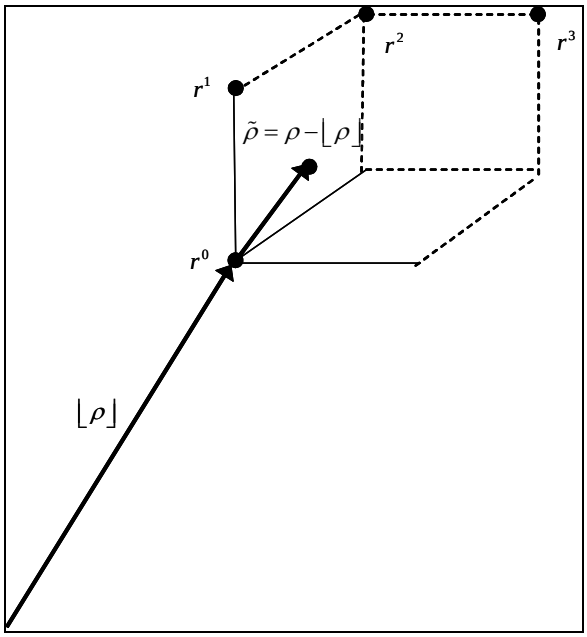

Figure 15. We can find a selection set for the fractional part of our continuous solution, without loss of generality.

7.2.4.2 COMPUTE_GRADIENT

Given the set of points in $S(\rho)$, we can evaluate each configuration represented using the PA to find a set of QoS estimates $\{Q_d(r^i)\}_{i=0}^{N}$. We can use those estimates to calculate a gradient $\nabla Q_c(\rho)$ in the following fashion.

```
compute_gradient( {Q_d(r^i)}_{i=0}^{|R|} ) :
for i = 1..|R|
        ∇Q_c(ρ)_i = Q_d(r^i) − Q_d(r^j)
end
```

Figure 16. Pseudocode for computing the gradient estimate.

Note the following fact about $S(\rho)$: if $r^i \in S(\rho)$, then there exists $r^j \in S(\rho)$ such that $r^i - r^j = e_i$ (that is, there is a vector that differs from r^i in the *i*th component). It is sufficient to order the vectors in $S(\rho)$ lexicographically to discover which vectors to use in the differences above.

In words, the *i*th component of the gradient vector is the difference between the PA QoS of the *i*th configuration in $S(\rho)$ and the PA QoS of the configuration in $S(\rho)$ that differs by one in the *i*th component.

7.2.4.3 ENFORCE_CONTINUOUS_FEASIBILITY

The next candidate continuous solution reached by a step in the steepest ascent must be projected onto the continuous constraint set to yield a closest feasible continuous solution.

- For the activation subproblem, all components must be between 0 and 1. Thus, any negative component is replaced with 0 and any component greater than 1 is reduced to 1.
- For the installation subproblem, we must enforce that the transfers out of a resource class must not exceed the number of tokens in the "nominal" case. That is, considering the vector of installation variables T as a matrix, $\sum_j \tau_{ij} \le N_i$ for each j.

  We project τ_i onto its feasible region for each resource class *i* by proportionally reducing each nonzero component until the sum of its components meets the constraints:

  First, consider only nonzero components of τ_i. Let $\overline{\tau}_i$ be this vector of nonzero components.

  Let $\alpha = \min\{\overline{\tau}_{ij}\}$ for $j = 1..|R|$. Also, let $\beta = (\sum_j \overline{\tau}_{ij} - N_i) / \text{length}(\overline{\tau}_i)$.

  – If $\alpha < \beta$, then set $\overline{\tau}_{ij} = 0$, where *j* = arg min{ $\overline{\tau}_{ij}$ }. In words, we reduce that minimum component to zero. We then repeat the process—the new $\overline{\tau}_i$ will be shorter by one component. If there are no remaining nonzero components, then we are done, and the only feasible set of installs is no installs.
  – Otherwise, we can reduce each nonzero transfer in the continuous solution equally by β. So **for each** *j*, $\overline{\tau}_{ij} = \overline{\tau}_{ij} - \beta$. If we take this branch, we are done, and return with a feasible solution.

7.2.4.4 ENFORCE_INTEGRAL_FEASIBILITY

For both subproblems, once we have a feasible continuous solution that passes the stopping condition, we want to project it onto the closest feasible integer solution. It can be shown that the rounding the value of each variable in the continuous solution to the nearest integer produces the desired solution.

For the assignment subproblem, optimization variables must have the value 0 or 1. If we simply round off each component, we get the nearest feasible integer solution.

For the installation subproblem, the variables representing installations from a resource class must sum to be less than the nominal number of tokens in the resource class. It suffices to show that the rounding solution works when the continuous solution is such that a group of variables sums to be exactly the nominal number of tokens. In this case we are in an easy extension of the Lemma 3.1 situation in Gokbayrak and Cassandras.

7.2.5 CoA Search Restart Procedure

The hill-climbing search of Section 7.2.2 will only find a locally optimal cyber-CoA, reached from the starting solution of the immediate post-attack configuration. EBCOTE-3 can generate other locally optimal solutions for sysadmin consideration, time permitting. To find other locally optimal CoAs, we use the following algorithm for identifying new feasible restart points for the gradient-directed search:

1. For each resource class having no tokens in it (an *empty* resource class) in the attack configuration, identify possible *source* resource classes for the empty class from which tokens may be installed. Source classes may not be empty.

   The source classes for empty class *E* may be identified by looking at the install matrix. The source classes are rows representing non-empty classes with non-dash entries in *E*'s column.

2. Construct a list of redistribution options that take tokens from source classes and install them to empty classes. The list is constructed as follows:

   Step 1 gives a list of possible source resource classes S_{E_i} for each empty class E_i. The list of redistribution options is the set of elements in the Cartesian product $\times_i S_{E_i}$, representing a choice of a single source class for each empty class.

   COA Search will only restart MAX times, where MAX is a user-defined constant, thus the exhaustive list of redistribution options need not be enumerated.

3. If the surrogate optimization terminates by finding a local maximum and MAX has not been reached, COA Search will restart using the next redistribution option. For each redistribution option:

4. For each source class *S* in the redistribution option

   - Let *k* = (# of initial tokens in *S* - 1)/(number of empty classes matched with *S* in the option).
   - Let Distributed = $\lfloor k \rfloor$
   - If Distributed = 0, *discard* option.
   - Else install *k* tokens to the empty class.

   If the option is not discarded, execute surrogate optimization using the initial config generated in step 3, and increment the restart counter. NOTE : Discarded options are not counted against MAX

7.3 KNOWN ISSUES

In our test case evaluation, we identified several opportunities for improvements on the cyber-CoA search algorithm described here. The surrogate optimization approach is valuable for optimization problems with a certain characterization—not all workflow models yield such problems.

First, surrogate optimization is not the ideal approach for the activation subproblem. The activation subproblem is an optimization over binary variables representing assignment of resource specs

to tasks. The problem is limited to a unit hypercube and all solutions are on the cube's vertices. We adapted the gradient-based search for the subproblem to promote uniformity in the initial EBCOTE-3 prototype; however, there are better methods for search over sets of binary variables that can be implemented in future development.

Surrogate optimization is appropriate for problems whose continuous relaxations exhibit smooth variation in QoS at the scale of discrete problem solutions. Some workflows may not fit these conditions. We observed in some problems that local minima appeared rather densely in the continuous problem, meaning that sometimes the gradient computed at a continuous solution was not a good guide to improving movements in the discrete space. More research is required to identify and characterize workflows for which surrogate optimization is a convenient approach and those for which other search methods may be better.

SECTION 8

CONCLUSIONS

8.1 SUMMARY OF EBCOTE-3 RESULTS

The EBCOTE-3 project has developed a prototype tool for generating cyber-CoAs to help system administrators of BMC3 systems reconfigure system resources in response to cyber attacks. EBCOTE-3 features include:

- Works on a broad range of workflow models specified using a Petri Net workflow modeling tool.
- Uses a resource policy model declaring cyber-resources and their attributes, roles that qualify resources for workflow tasks, and installation times required to transfer resource tokens from one spec to another.
- Evaluates impacts of the current resource configuration and status on QoS.
- Computes a QoS estimate including user-specifiable priorities and deadlines for each job class.
- Accounts for any jobs currently in-progress.
- Searches for multiple cyber-CoAs within a user-specified time window and ranks their effectiveness estimates.
- Provides APIs to allow external WfMS programs to supply workflow data (such as job history logs) to EBCOTE-3.

EBCOTE-3 uses a novel path reconstruction analysis technique that propagates the effects cyber-attack or reconfiguration perturbations throughout a sample history of BMC3 workflow processing. This technique provides a factor of 6-7 speedup over workflow discrete event simulation, a benefit magnified by the optimal search algorithm which uses it.

EBCOTE-3 leverages path reconstruction to support an innovative surrogate optimization technique that searches for cyber-CoAs through a gradient-based search on a continuous relaxation of the BMC3 QoS optimization problem. Here, path reconstruction provides a fast method for computing the objective function values of various configurations, permitting an estimate of the gradient at some solution to the continuous problem. EBCOTE-3 finds cyber-CoA options which preserve QoS within seconds, developing improved CoAs within a couple of minutes if time permits.

Surrogate optimization is not ideal for every workflow problem, but the path reconstruction analysis technique is suitable for supporting other stochastic discrete optimization methods which use an oracle to guide search. More work is needed to characterize workflow problems and identify the most advantageous search mechanism for each.

8.2 FUTURE DIRECTIONS

One area for future work on the EBCOTE prototype is further integration with evolving WfMS standards. In particular, EBCOTE-3 relies on specified timing rule distributions for each task in the workflow to perform path reconstruction analysis. Each task in the workflow model must have a declared distribution—one of constant, uniform, exponential, triangular, or normal. Further, the dependence of these distributions on resource token attributes must be specified. This information does not currently appear in workflow definitions standards for WfMS. EBCOTE needs a methodology for inferring properties of task distributions from data in the history log.

The EBCOTE prototype can also benefit from additional research into search algorithms. For instance, as discussed in Section 7.3, searches other than surrogate optimization are more appropriate for the activation subproblem, and would likely improve performance.

EBCOTE searches for cyber-CoAs that specifically reconfigure resource usage in the BMC3 system, but not only may system administrators wish to use other types of workflow configurations, but more sophisticated cyber-attacks may alter other features of the workflow. Path reconstruction analysis can be applied to other workflow perturbations such as decision rules and local queue management rules.

We believe that the application of path reconstruction techniques to workflow management opens up many possibilities for workflow and resource optimization, benefiting business process re-engineering, infrastructure protection, and defense BMC3 communities. EBCOTE is an important step for cultivating these ideas and adding value to BMC3 cyber-protection.

REFERENCES

[1] van der Aalst, W. and van Hee, K., *Workflow Management: Models, Methods, and Systems*, MIT Press, 2002.

[2] Alhusaini, A., Prasanna, V., Raghavendra, C., "A Unified Resource Scheduling Framework for Heterogeneous Computing Environments", *8th Heterogeneous Computing Wksp.: HCW '99* (San Juan, Puerto Rico, 12 Apr 1999), p. 156-165.

[3] ALPHATECH, Inc., "ALPHA/Sim User's Guide", Burlington, MA, 1998.

[4] ALPHATECH, Inc., "EBCOTE: Predicting the Impact of Disruptions to BMC3 Workflow", TR-1003 (EBCOTE-1 Final Report), Burlington, MA, 1998.

[5] Andradottir, S., "A Review of Simulation Optimization Techniques", in *Proceedings of the 1998 Winter Simulation Conference*, 1998.

[6] Cardoso, J. and A. Sheth (2002). "Implementing QoS Management for Workflow Systems." *International Journal of Cooperative Information Systems (IJCIS)*, submitted**.**

[7] Cardoso, J., A. Sheth and J. Miller (2002). "Workflow Quality of Service". *International Conference on Enterprise Integration and Modeling Technology and International Enterprise Modeling Conference (ICEIMT/IEMC'02)*, Valencia, Spain, Kluwer Publishers.

[8] Cassandras, C.G., *Discrete Event Systems*, 1993.

[9] Du, W.; Davis, J.; Huang, Y.; Shan, M., "Enterprise Workflow Resource Management", Hewlett Packard Labs Technical Report, HPL-1999-8.

[10] Firoiu, V., and Towsley, D., "Call Admission and Resource Reservation for Multicast Sessions", Univ. of Massachusetts Technical Report UM-CS-1995-017, revised 1996.

[11] Gertphol, S., Yu, Y., Gundala, S., Prasanna, V., Ali, S., Kim, J., Maciejewski, A., and Siegel, H., "A Metric and Mixed-Integer-Programming-Based Approach for Resource Allocation in Dynamic Real-Time Systems", In *Proceedings of International the International Parallel and Distributed Processing Symposium*, April 2002.

[12] Gokbayrak, K., and Cassandras, C.G., "A Generalized 'Surrogate Problem' Methodology for On-Line Stochastic Discrete Optimization", *J. of Optimization Theory and Applic.*, Vol. 114, 1, pp. 97-132, 2002.

[13] Huang, Y., and Shan, M., "Policies in a Resource Manager of Workflow Systems: Modeling, Enforcement and Management", Hewlett Packard Labs Technical Report, HPL-98-156.

[14] Ho, Y. and Cao, X., *Perturbation Analysis of Discrete Event Dynamic Systems*. Norwell, MA: Kluwer, 1991.

[15] Klingemann, J., "Controlled Flexibility in Workflow Management", *Proceedings of the 12th International Conference on Advanced Information Systems (CAiSE '00)*, Stockholm, Sweden, Springer-Verlag, 2000.

[16] Ramamoorthi, R., Rifkin, A., Dimitrov, B., and Chandy, K., "A General Resource Reservation Framework for Scientific Computing", *Proceedings of the First International Scientific Computing in Object-Oriented Parallel Environments (ISCOPE) Conference*, Marina del Rey, December 1997.

[17] Robinson, A and Lounsbury, D. "Measuring & Managing End-to-End Quality of Service (QoS) Provided by Linked Chains of Application and Communications Services," www.opengroup.org, 2001.

[18] Workflow Management Coalition, WfMC Terminology and Glossary, WfMC-TC-1011, February 1999.

[19] www.htc.honeywell.com/projects/arm

[20] www.wfmc.org/standards/standards.htm