January 2016

A Localized Autonomous Control Algorithm For Robots With Heterogeneous Capabilities In A Multi-Tier Architecture

Jeremy Straub

A LOCALIZED AUTONOMOUS CONTROL ALGORITHM FOR ROBOTS WITH HETEROGENEOUS CAPABILITIES IN A MULTI-TIER ARCHITECTURE

by

Jeremy Straub
Master of Business of Administration, Mississippi State University, 2010
Master of Science, Jacksonville State University, 2011

A Dissertation

Submitted to the Graduate Faculty

of the

University of North Dakota

In partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Grand Forks, North Dakota
May
2016

Elements of this document relate to the control of cyber-physical systems in a research environment. Users are expressly cautioned that the performance of algorithms and systems in a real world environment will necessarily differ somewhat from the simulation-based work presented herein. The Author, the University of North Dakota and their successors in interest, affiliates and other related parties shall bear no liability for the use of this work in any real world, mission critical or other environment. The user assumes all risk of reliance on this work and is explicitly encouraged to exercise due diligence in this regard.

This dissertation, submitted by Jeremy Straub in partial fulfillment of the requirements for the Degree of Doctor of Philosophy from the University of North Dakota, has been read by the Faculty Advisory Committee under whom the work has been done, and is hereby approved.

Ronald Marsh, Ph.D.

David Whalen, Ph.D.

Hassan Reza, Ph.D.

Travis Desell, Ph.D.

William Semke, Ph.D.

This dissertation is being submitted by the appointed advisory committee as having met all of the requirements of the Graduate School at the University of North Dakota and is hereby approved.

Wayne Swisher
Dean of the Graduate School

Date

Title A Localized Autonomous Control Algorithm for Robots with Heterogeneous Capabilities in a Multi-Tier Architecture

Department Computer Science

Degree Doctor of Philosophy

In presenting this dissertation in partial fulfillment of the requirements for a graduate degree from the University of North Dakota, I agree that the library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by the professor who supervised my dissertation work or, in his absence, by the Chairperson of the department or the dean of the Graduate School. It is understood that any copying or publication or other use of this dissertation or part thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of North Dakota in any scholarly use which may be made of any material in my dissertation.

Jeremy Straub
April 25, 2016

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Table Page

ACKNOWLEDGEMENTS

This work has been supported by a Grant-In-Aid of Research from Sigma Xi, The Scientific Research Society, North Dakota EPSCoR (NSF # EPS-0814442) and a Summer Doctoral Fellowship from University of North Dakota School of Graduate Studies.

Thanks is given to the members of my dissertation committee.

This dissertation builds on and draws upon other prior work of the author. These works may include, but are not limited to:

- Straub, J., H. Reza. 2015. A Blackboard-Style Decision Making System for Multi-Tier Craft Control and its Evaluation. Journal of Experimental & Theoretical Artificial Intelligence, Vol. 27, No. 6.
- Straub, J. 2014. Command of a Multi-Tier Robotic Network with Local Decision Making Capabilities. International Journal of Space Science and Engineering, Vol. 2, No. 3.
- Straub, J. 2016. Cybersecurity Methodology for a Multi-Tier Mission and Its Application to Multiple Mission Paradigms. Proceedings of the 2016 IEEE Aerospace Conference.
- Straub, J. 2016. The Development of a Simulation Environment for Testing of a Multi-Tier Mission Command Architecture. Proceedings of the 2016 IEEE Aerospace Conference.
- Straub, J. 2014. Building Space Operations Resiliency with a Multi-Tier Mission Architecture. Proceedings of the SPIE Defense + Security Conference.
- Straub, J. 2013. A Data Collection Decision-Making Framework for a Multi-Tier Collaboration of Heterogeneous Orbital, Aerial and Ground Craft. Proceedings of the SPIE Defense, Security + Sensing Conference.
- Straub, J. 2013. Integrating Model-Based Transmission Reduction into a Multi-Tier Architecture. Proceedings of the 2013 IEEE Aerospace Conference.
- Straub, J. and R. Fevig. 2012. Multi-Tier Planetary Exploration: A New Autonomous Control Paradigm. Proceedings of the AIAA Space 2012 Conference.
- Straub, J. 2012. Multi-Tier Exploration: An Architecture for Dramatically Increasing Mission ROI. Proceedings of the AIAA Space 2012 Conference.
- Straub, J. 2012. Multi-Tier Exploration Concept Demonstration Mission. Proceedings of the 2012 Global Space Exploration Conference.
- Straub, J. 2014. The Critical Role of CubeSat Spacecraft in A Multi-Tier Mission for Mars Exploration. Presented at the Mars CubeSat/NanoSat Workshop.
- Straub, J. 2013. The Multi-Tier Mission Architecture and a Different Approach to Entry, Descent and Landing. Presented at the International Planetary Probe Workshop.
- Straub, J. 2012. A Role for Cubesats in a Multi-Tier Exploration / Reconnaissance Architecture. Presented at the 9th Annual Cubesat Workshop.
- Straub, J. 2012. Cubesat Integration into a Multi-Tier Exploration Framework. The 2012 European Cubesat Symposium.
- Straub, J., R. Marsh. 2015. A Comparison of Centralized and Decentralized Blackboard Architecture-Based Command Techniques for Robotic Control Under Varying Conditions. Submitted to Expert Systems with Applications.
- Straub, J. 2013. Automating Maintenance for a One-Way Transmitting Blackboard System and Other Purposes. Accepted for publication in Expert Systems.
- Straub, J. 2013. A Distributed Blackboard Approach Based Upon a Boundary Rule Concept. Journal of Intelligent & Robotic Systems (in press, initial online publication Sept. 30, 2015).
- Straub, J. 2015. Comparing the Effect of Pruning on a Best-Path and Naïve-Approach Blackboard Solver. International Journal of Automation and Computing, Vol. 12, No. 5.
- Straub, J. 2014. Evaluation of a Multi-Goal Solver for Use in a Blackboard Architecture. International Journal of Decision Support System Technology, Vol. 6, No. 1.
- Straub, J. 2015. Using Swarm Intelligence, a Blackboard Architecture and Local Decision Making for Spacecraft Command. Proceedings of the 2015 IEEE Aerospace Conference.
- Straub, J. 2014. Comparing the Blackboard Architecture and Intelligent Water Drops for Spacecraft Cluster Control. Proceedings of the AIAA Space 2014 Conference.
- Straub, J., H. Reza. 2014. The Use of the Blackboard Architecture for a Decision Making System for the Control of Craft with Various Actuator and Movement Capabilities. Proceedings of the International Conference on Information Technology: New Generations.

ABSTRACT

This dissertation makes two contributions to the use of the Blackboard Architecture for command. The use of boundary nodes for data abstraction is introduced and the use of a solver-based blackboard system with pruning is proposed. It also makes contributions advancing the engineering design process in the area of command system selection for heterogeneous robotic systems. It presents and analyzes data informing decision making between centralized and distributed command systems and also characterizes the efficacy of pruning across different experimental scenarios, demonstrating when it is effective or not. Finally, it demonstrates the operations of the system, raising the technology readiness level (TRL) of the technology towards a level suitable for actual mission use.

The context for this work is a multi-tier mission architecture, based on prior work by Fink on a "tier scalable" architecture. This work took a top-down approach where the superior tiers (in terms of scope of visibility) send specific commands to craft in lower tiers. While benefitting from the use of a large centralized processing center, this approach is limited in responding to failures and interference.

The work presented herein has involved developing and comparatively characterizing centralized and decentralized (where superior nodes provide information and goals to the lower-level craft, but decisions are made locally) Blackboard Architecture based command systems. Blackboard Architecture advancements (a solver, pruning, boundary nodes) have been made and tested under multiple experimental conditions.

CHAPTER I

INTRODUCTION[1]

Fink [1, 2] and others [3, 4] have proposed the use of teams of multiple robots for exploring planets and other applications. These multi-robot teams generally require robots of multiple configurations. Under Fink's mission architecture, robots are separated in to tiers based on their scope of influence and movement characteristics: specifically, orbital, flying and ground-based tiers. Each tier exerts influence over craft in tiers of lesser range. As part of the characterization of the benefits and drawbacks of distributed and centralized control, a distributed approach is proposed and analyzed herein. Under this approach, control decisions are made locally, based on assigned goals. The higher-range tiers also have a role in the transmission and prioritization of data from the lower-range tiers and may deploy (and re-deploy) the lower-tier vehicles. This chapter provides an overview of this proposed control system, its control methodology, how it operates, the key planning and control module, and system intra-communications. These topics are expanded upon in subsequent chapters.

System Overview

The multi-tier, multi-craft control system must be able to effectively delegate

[1] This chapter is derived from: Straub, J. (2012), Multi-Tier Exploration Concept Demonstration Mission. Proceedings of the 2012 Global Space Exploration Conference and Straub, J. (2013), Control of a Multi-Tier Robotic Network with Local Decision Making Capabilities. Submitted to the Journal of Sensor and Actuator Networks.

decision making while ensuring craft coordination in working on complex goals. A multi-tier distributed management system is proposed which incorporates the concept of decision-making delegation and management by exception. Like a well-implemented human management system, each role is not attached to a specific craft. A role is assigned to a craft but is automatically reassigned if the craft is unable to carry it out. Generally, leader roles are assigned to craft based on their computational capabilities, visibility of and visibility to the group of craft that they manage. However, aside from communications constraints, there is no requirement for any particular assignment.

Multi-Tier Control Methodology

The proposed control methodology combines four key principles. First, the participating craft are organized hierarchically. Each craft has one superior (the primary orbital craft's superior is the ground controllers) and may have multiple subordinate craft. Second, goals are delegated from super craft to subordinate craft. The subordinate craft are responsible for meeting the requirements encapsulated within the goal message and/or advising if a goal is not achievable or completion criteria (such as a required timeframe) will be violated. Each individual craft, third, makes its own planning and scheduling decisions based on the combination of local constraints (e.g., power and other resource availability), local conditions (e.g., movement speed on local terrain) and delegated goals. Finally, a craft can task processing to (or request resources or assistance from) another craft that is better equipped, if needed. Three of these elements, goal delegation, local decision making and the utilization of resources from other craft, are now discussed.

Goal Delegation

High-level goals are assigned to the collection of craft by mission controllers. The primary craft creates a plan for carrying out the mission by decomposing goals into sub-goals which are delegated to collections of subordinate craft. Figure 1 depicts this decomposition for a conceptually simple task of conducting an exhaustive survey of a region. In this example, the craft are presumed to be homogeneous and equally distributed. A single orbital craft delegates the survey of three grid locations (that are part of region one) to three UAVs which each delegate the survey of six grid sectors (A-F) to their subordinate ground craft. In this case, it is presumed that each grid sector must be surveyed by a ground craft. This, however, is an atypical application for a multi-tier mission. The value of the multi-tier architecture generally comes from the intelligent use of assets. Specifically, in this case, by avoiding surveying regions at higher resolutions that are deemed to be insufficiently interesting, based on lower-resolution data.

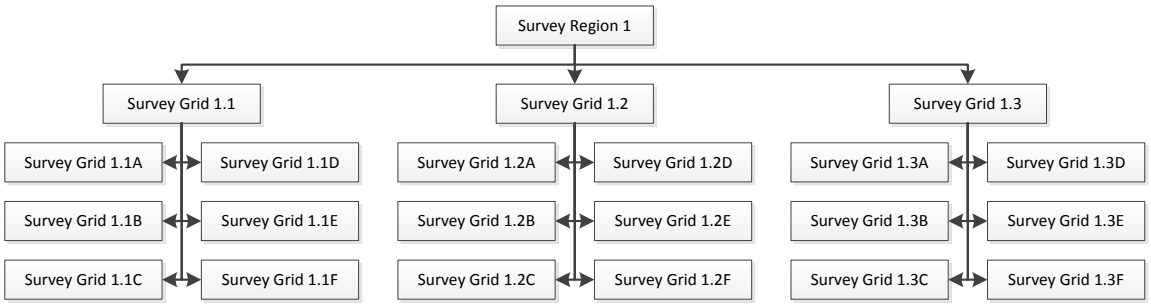

Figure 1. Decomposition of an Exhaustive Survey Task.

Another scenario is presented, in Figure 2, to illustrate this. In this scenario, only areas with features of interest are explored to higher levels of resolution. All three UAVs are dispatched, as the orbital spacecraft identified features of interest in three locations. However, the UAVs do not identify as many sub-goals for delegation to their subordinate craft, as certain regions are deemed insufficiently interesting to merit ground exploration.

This adaptive approach conserves resources and allows craft to be devoted to as high-value tasks as available. Note that in the scenario presented in Figure 2, survey locations could be divided between craft or defined differently to assign work to all craft for faster completion. The non-tasking shown in Figure 2 is designed to be illustrative of the difference as compared to Figure 1's exhaustive search approach, instead of a typical approach to problem solving. However, it would be indicative of a tasking scenario if the craft were assigned to other tasks or temporarily assigned to another group.

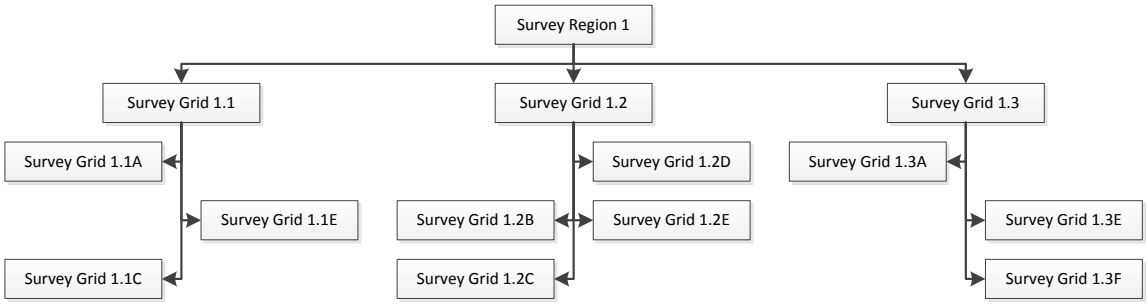

Figure 2. Decomposition of an Interest-Based Survey Task.

Local Decision Making

Goals are assigned via the hierarchical structure, but planning and scheduling for each craft and its subordinates is performed locally. This allows each craft to consider its local conditions and reported and derived (e.g., from task performance) subordinate conditions in determining how to best achieve delegated goals. Figure 3 depicts the decision making process for a craft with subordinates. First, it decomposes the assigned goal into component goals whose achievement results in the goal's achievement. For each, the craft determines whether it should work on the goal itself and/or delegate it. For those it will perform, it decomposes the goal into tasks and orders them within the goal and relative to other pending tasks. For sub-goals that are delegated, subordinate performance and condition information is used to determine goal assignment.

This process is continuous. For example, in the survey described previously (Figure 2), the first sub-goal (for the UAV-level craft) was for the craft itself to conduct an initial survey. From this, additional sub-goals (the ground surveys) were identified and tasked. The craft also re-assesses task ordering and subordinate assignment when assumptions (relied upon information from local conditions, global knowledge, subordinate conditions and subordinate performance) are invalidated or violated. The multi-tier model practices management by exception, where performance boundaries (both positive and negative) are identified. Violation of these boundaries triggers an autonomous investigation into its cause (e.g., an invalidated or violated assumption).

Utilization of Resources from Other Craft

One key advantage of the top-down model proposed by [1, 2] is the fact that the majority of processing is carried out on the most capable computer in the collection of craft (generally, on the orbital spacecraft). In the top-down model, this occurs because most decisions (and thus the supporting analysis) are made at this node. However, in many cases the benefits of local decision making and the benefits of utilizing the highest-performance computer for computation can be enjoyed concurrently.

Similarly, some tests require the coordination of several craft (e.g., lifting a heavy item or if multiple sensor capabilities are required). To service these needs, a request message is used to ask other craft for assistance. The sending craft provides a request prioritization, in terms of global evaluation metrics. The receiving craft compares this prioritization to other items in its goals and tasks lists and prioritizes it appropriately (negotiating with the requestor regarding timing, if concurrent action is required).

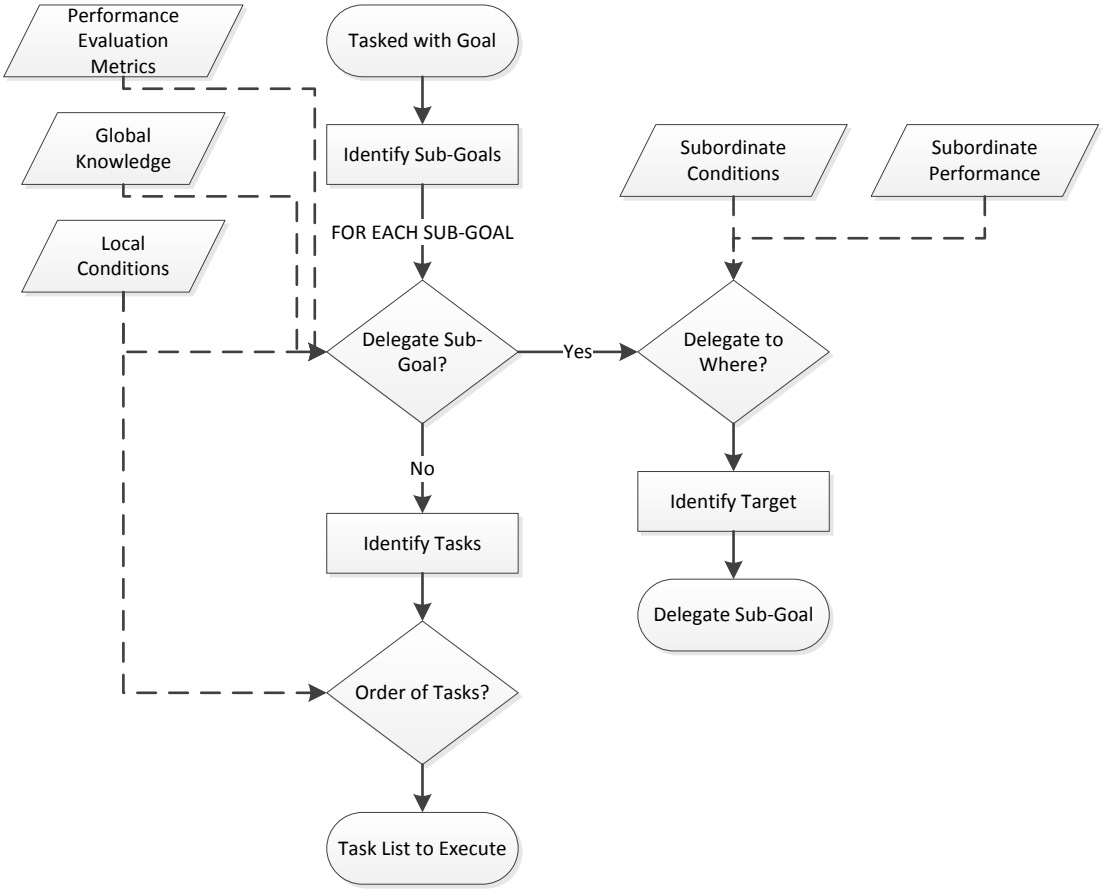

Figure 3. Local Decision Making Process for Craft with Subordinate Craft.

Combined Operations

Each control program operates in a waiting loop state. Local and group control routines share the computational resources of the group leader craft. Action is driven by interrupts; each triggering condition is evaluated and either immediately acted upon or queued for later action. Each request (running and queued) is assigned a priority; any incoming request of higher priority overrides the current request being processed. Request priority is based on the combination of task priority and suitability metrics (closeness, equipment suitability), as determined by the analysis module. Modules commanding complex (and/or perilous) maneuvers can temporarily suspend interrupt processing to

ensure that no intervening request causes maneuver failure. Additionally, running requests receive a priority boost to avoid the interruption of operations which would have to be reattempted later to process a marginally more important request.

If no other higher-priority action is tasked to the craft, random track exploration is performed. Exploration is only undertaken, however, subject to power usage and other operating constraints. Craft with a fixed and non-renewable fuel source (that would be consumed by this exploration) are generally excluded from random track exploration.

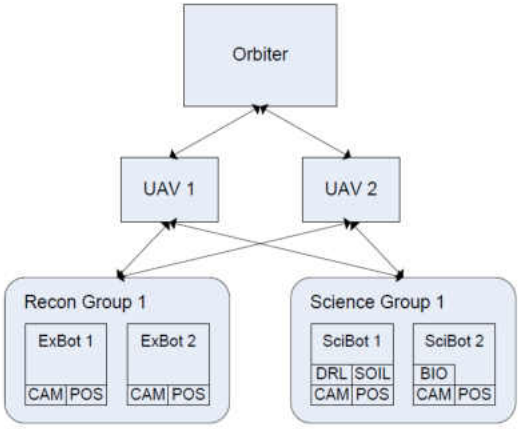

Figure 4. Example mission architecture.

Group Leaders

The top of the hierarchy is filled by a leadership node (identified as 'Orbiter' in Figure 4). This node is a super group leader, as the scope of its group is the entire mission. Its upstream communications are with the human or automated controller. Aside from these two differences, the leadership node is simply a group leader.

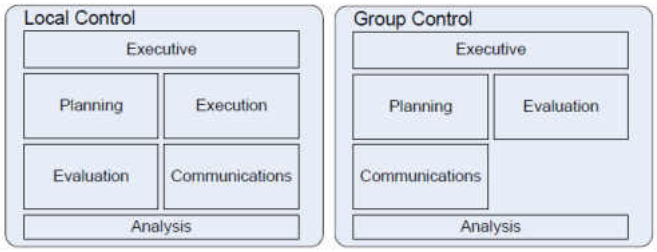

Figure 5. Local and group control diagrams.

The control module for group leader nodes (shown in Figure 5, right) has five component modules: executive, planning, evaluation, communications and analysis. The executive module is responsible for managing compliance with storage and other constraints. It is also responsible for assuming emergency control (based on rules, general objectives and assigned tasks) if upstream communications are disrupted.

The planning module evaluates goals relative to data, assertions and rules on the local blackboard (some of which will have originated from the blackboards of superior and inferior nodes) and delegates sub-goals to group members or subgroups. Weighted proximity (based on the cost of the estimated path of travel), suitability (based on sensor configuration) and task compatibility (based on other currently assigned goals and craft sensor/actuator availability) are used to make delegation decisions.

The evaluation module reviews progress. It identifies goals that have reached an exception condition (e.g., insufficient progress based on time or resources consumed) for review and resolution. It also identifies lessons learned from completed and in-progress tasks (e.g., updated cost and time values for task types) for use in future planning.

The communications module is responsible for maintaining contact with upstream and downstream communications partners. The communications module is also responsible for scheduling communications based on relative priority and applicable constraints when requests upon the system exceed capacity.

Worker Nodes

A node that has no subordinates (e.g., ExBots 1 and 2 and SciBots 1 and 2 in Figure 4) is a worker node. Worker nodes perform the tasks necessary to achieve their assigned goals autonomously and report upon completion or encountering an exception-condition.

The worker control module (Figure 5, left) has six component modules: executive, planning, execution, evaluation, communications and analysis. Worker component modules perform substantively the same as the group-level similarly named modules, except with a local scope. The execution module is responsible for generating commands and transmitting them to lower level hardware control systems.

Craft Control

A craft-specific control system translates each task in to a completion plan and commands to effect task completion, which are translated to low-level commands that are sent to hardware controllers. Each craft also has a data analysis module to identify and prioritize data that should be placed on the blackboard of higher levels of the hierarchy. While low level control routines and action control systems vary considerably between craft types, the structure of the control system is consistent framework-wide.

Analysis, Planning and Tasking

The Central Analysis, Planning and Tasking (CAPaT) system is the overall director of the long-term operations of the mission group. It runs on the leadership node, but can run on alternate craft, if the original node is incapacitated.

Analysis & Target Identification

The leadership node CAPaT module creates sub-goals, based on mission goals, which are communicated to group leader and worker node planning modules. Each goal is comprised of a priority and one or more rules which, if executed, constitute the goal's

satisfaction. Group leader planning modules store all active priorities for goals within their scope of command. Worker nodes store only goals that are applicable to the craft.

The executive on each craft forwards sensed data to the analysis module for identification, rule matching and transmission prioritization. The analysis module supplies the executive with transmission priority (or unworthy of transmission classification) and data to send to the superior node's blackboard. Collected data is placed on the local blackboard by analysis module, possibly triggering planning changes.

Planning & Tasking

The group leader planning module is responsible for plan generation for all subordinate groups and craft. It generates a long-range plan based on current mission goals and delegates sub-goals to each of its subordinates. Weighted task prioritization and cost minimization values are used for goal ordering. Planning and tasking threshold values determine how far in to the future the node plans and communicates plans to subordinates, respectively. At the worker node level, the planning module is responsible for generating plans for task completion. This includes determining the target visitation order, the travel path, and the sensing schedule. It has primary responsibility for constraint compliance and combines global task estimates (refined over time) and local correction values to generate an estimated schedule (inclusive of an error margin).

Communications Control & Planning

The communications control and planning module is responsible for scheduling and operating the communications subsystem based on upstream and downstream transmission

priorities. In group leaders, this system receives internal and subordinate requests for upstream communications and downstream requests directed at its own and subordinate control systems. The module calculates communications schedules (based on communications partner availability, see [5]) for transmissions. Further, it advises subordinate craft as to its availability for routine communications. It deals with both upstream and downstream emergency communications in real time and adjusts the communications schedule. At the worker level, the communications system maintains upstream node availability schedules and general priority level information (to prevent sending data that will be discarded due to its low priority). The communication system accepts prioritized data and other messages from the executive and generates and executes a transmission plan.

System Communications

System communications are based on the philosophy of management by exception [6, 7] and data transmission by priority [8-10]. Downstream messages include goal delegation, task time estimate updates and blackboard updates. Upstream operational messages include blackboard data, completion and exception notifications. Upstream communications also include responses to poll requests for task time average calculation.

Summary

The remainder of this document provides more details on the above presented topics. Chapter II provides an overview of prior work. Chapter III discusses system implementation and operations in greater detail. Chapter IV presents the experimental

design and methodology utilized. Chapter V presents and analyzes the experimental results. Finally, Chapter VI concludes the paper and discusses directions for future work.

CHAPTER II

BACKGROUND[2]

The work presented herein draws from multiple research areas. Fink's work [1, 2] defines the concept of a multi-tier mission providing craft-role and tier-level definitions. It [1] also discusses data collection prioritization in a multi-tier environment. Sensor-web research (e.g., [11-14]) suggests multiple ways of coordinating sensing element collections to achieve science goals. Centralized control, bidding-based decentralized control, and collaborative team-based approaches are discussed. Work on robotic control (e.g., [15-18]) provides a basis for group organization and craft operation. Ground position identification techniques, without using positioning satellites, are discussed by [19-21] and remain an active research topic. UAV autonomous navigation work (e.g., [22, 23]) provides a foundation for aerial tier autonomous flight control.

Autonomous Robotics

An understanding of the types of robots that would be controlled as part of a multi-tier system informs control decisions. Applications of orbital robot autonomy include spacecraft docking (the Soviets with IGLA and KURS [24] and the United States with ASTRO and NextSat [25]). Planning for orbital craft was demonstrated by DS-1's Remote

[2] This chapter is derived from: Straub, Jeremy. 2011. A Review of Spacecraft AI Control Systems. In the Proceedings of the 15th World Multi-Conference on Systemics, Cybernetics and Informatics.

Agent Experiment [26] and EO-1's CASPER mission planning software [27]. Health status assessment and repair was demonstrated with DS-1's MIR system [28] and EO-1's Livingstone Version 2 software [29]. Command software (AutoNav on DS-1 [30] and software on Hayabusa [31], Rosetta [32] and Deep Impact's impactor [33, 34]) has also been demonstrated. These systems have lowered human staff requirements: DS-1 required significantly less than the 100 to 300 staff required for Cassini [35], for example, using a beacon methodology [30] (requesting aid only when required) freeing the Deep Space Network [36] or allowing more science data to be transmitted [35].

Significant prior work has been performed on the control of unmanned aerial vehicles. Schlecht, et al. [37] show how it can be done using only localized commuications. Lua, et al. [38] discuss swarm-style techniques for performing a task with minimal communications. Schesvold, et al. [39] use a partially observable Markov process for planning, pitting short term against possible longer-term greater gain. Control of very small UAVs, micro-aerial vehicles (MAVs) in a lozalized environment is discussed by Michael, Stump and Mohta [40], who utilize a central system manager and solver, which implements blackboard-like principles.

In surface robotics, a variety of control techniques have been considered. Punzo, et al. [41] present a swarm-based small autonomous robot planetary exploration approach. Ambler used terrain maps including elevation and uncertainty data [42, 43] and made decisions based on goal comparison and craft capability self-awareness [43]. The Self-Mobile Space Manipulator (a robotic service arm) used neural networks for control [44]. Dante I's autonomous control software operated by sensing, planning and then acting [45]: an operator supplied trajectory was validated and then executed. Dante II, instead of

relying on terrain data, used servo mechanism feedback to control its walking motion [46, 47]. Rocky 7 demonstrated autonomous navigation based on controller-supplied waypoints [48]. NOMAD used image processing of onboard camera data for obstacle detection and terrain classification [49], creating its own traversal suitability map for both desert [49] and polar [50] traversals. Hyperion demonstrated sun-synchronous navigation with sliding autonomy ranging from teleoperation to full autonomy [51]: 90% of its travel was able to be conducted autonomously [52].

Zoe's [53, 54] science planner, science observer, instrument manager and instrument controller components and combined satellite and local imagery [55], using an optimistic planning approach. Scarab [56] demonstrated autonomous navigation based on a static three-dimensional point cloud model. For Sojourner [57], on the other hand, control was autonomous but planning was done on Earth [58]. The Spirit and Opportunity rovers' use of autonomous driving significantly increased their movement speed [59], by allowing the rover to navigate based on a wide-area terrain map [60]. Imagery is also used to determine travel distance and to correct for slippage [59]. Human rover ground planning is done with MAPGEN software [61].

Control of Robotic Systems

Individual components have been discussed. Now, focus turns to various methods for controlling collections of robots. Prior work in this area is now presented.

The Automated Scheduling and Planning Environment (ASPEN), an artificial intelligence-based scheduling and planning system, breaks down goals in to a sequence of commands to send to a spacecraft [62]. It models spacecraft in terms of activities,

parameters and associated dependencies, temporal, resource, state variable and reservations constraints [63]. It looks at scheduling from a repair perspective: identifying and fixing constraint violations. An iterative repair algorithm, which uses heuristics with associated confidence levels to order violation correction attempts, is used [63].

The Distributed Robotic Architectures (DIRA) project created a framework for coordinating collections of robots [64]. A three-layer system where each layer communicates with its corresponding layer in other robots was developed. The planner breaks down goals, creates plans and coordinates teams and commitments. The executive layer runs plans and communicates with other executive layers for coordination. The behavior layer provides reactive control and coordinates group physical interaction.

The CASPER continuous planning system [65] extends ASPEN, adding dynamic planning and scheduling capabilities [66]. It has a modeling language, constraint management system, search and repair heuristics, and a temporal constraint management system. It continuously updates plans based on real-time activity, system state and resource information, making the system responsive to changing conditions [67].

The Closed Loop Execution and Recovery framework combines a planner's global perspective with a reactive executive's responsiveness. It strikes a balance between non-replenishable resource management and reactiveness [68].

OASIS [69] autonomously analyzes rover data, prioritizing it by interest level. It also identifies exploration opportunities and has planning and scheduling components.

The Modified Antarctic Mapping Mission [70] had a four step planning process consisting of selecting swaths which provide coverage of the desired area, creating a collection schedule, creating a downlink schedule and validating the schedule's constraint

and goal compliance. The mission demonstrated "overwhelmingly successful" automation, lowered costs and increased science return.

The TEMPEST planning system uses terrain, solar visibility, Earth visibility and vehicle state information for planning [51]. It is able to replan using an algorithm which by propagates changes to only affected areas. It has deliberative and functional layers.

Unmanned air, ground and surface vehicles are being developed by the U.S. Army and Navy MDARS program, the U.S. Army's Future Combat System (FCS) program, the DARPA's PerceptOR program, the COUGAR program, and the U.S. Army and Navy SPARTAN Advanced Concept Technology Demonstration program [71].

MDARS and PerceptOR are ground vehicles which can serve as a mobile launch, landing and support platform for UAV units. SPARTAN is a water-based vehicle which can serve as a UAV base. The FCS program incorporates UAVs as part of a network-centric combat system. The COUGAR system has a command vehicle, long range weapons robot, and UAV. The UAV surveys targets and confirms the missile strike. All of these currently require some level of human control.

The Hetereogeneous Agricultural Research Via Interactive, Scalable Technology project (HARVIST) is an intelligent system for combining multiple data sources to make predictions about crop yield. These include satellite imagery and weather data used [72].

Sensorwebs, node networks which take action based on the detection of an event-of-interest [73], are being implemented for various purposes [74]. For example, a volcano sensorweb may detect an eruption with an in-volcano sensor or low resolution orbital satellite. Based on this, the sensorweb requests observation from a planning service which evaluates it and forwards it to a satellite for high-resolution imagery. The onboard planner

evaluates the request and takes the requested actions, if possible [75].

Blackboard Architecture

The MTAMA utilizes a Blackboard-style architecture. The Blackboard Architecture utilizes a set of rules, facts and actions for decision making. Facts represent knowledge (and can either be asserted or not) about the environment (or other matters). Actions are, as the name suggests, activities that the system can perform or have performed. Rules interconnect the system. A rule is triggered by having its pre-conditions met and it can assert one or more facts and/or trigger one or more actions. Focus now turns to prior work on Blackboard architectures and their use in robotic control.

In [76], Hayes-Roth presents the Blackboard architecture, an enhancement of the Hearsay-II system [77]. The architecture functions like an expert system (e.g., [78, 79]) which triggers actions instead of making recommendations. It is comprised of two blackboards (for domain and control problems). Problem solutions are arrived at by triggering rules on the blackboard. When new information is added to the blackboard, all rules whose activation conditions are satisfied are placed in the "Invocable-List".

An activated rule is selected based on its rating and priority. It can create events or modify the system state triggering other rules and/or actions. Once a rule has executed, a cycle of assessing and selecting an activated rule continues until a solution is found or no activated rules exist. The architecture provides documentation capabilities, as each rule created, activated or modified and each action is recorded.

Numerous applications have demonstrated the Blackboard concept. The PROTEAN system [80] models protein structures. It operates on top of ACCORD which

provides a conceptual network creation mechanism, vocabulary, a hierarchy representation mechanism and template set for representing actions, states and events.

The SRI Procedural Reasoning System [81] is designed to solve the dual need of attaining larger goals while reacting to environmental changes in real time. The primary contribution of this work is the notion of running multiple blackboard-like structures concurrently (running asynchronously and utilizing message passing to communicate).

Rice [82] presents Poligon, a language for implementing applications which follow the Blackboard problem-solving model. It provides a syntax and framework for the creation of a Blackboard-architecture-based system. Corkill, Gallagher and Johnson [83] created an abstraction model to resolve the issue of implementations either being haphazard, maximizing efficiency at the expense of flexibility or maximizing flexibility at the expense of efficiency. Le Mentec and Brunessaux [84] modified Atome to create the Lisp-based Atome-tr, which reacts quickly to changes via parallel processing, an interrupt system and dynamic planning. It is comprised of the overall strategy, tasks, specialists and multiple blackboards with state information. Asynchronous updating and summary blackboards (containing subsets of relevant information) are also utilized.

Hewett and Hewett [85] contend that prior work on the Blackboard architectural approach had suffered due to a lack of a common language to facilitate comparison. They define a language comprised of four categories: actions, events conditions, state conditions and "context generators." All elements of the language are human-readable statements, generally resembling "ADD <object name> to <level-name>". To improve efficiency, they utilize a technique for knowledge computation, a network based on RETE for triggering and a "demon architecture" for task list maintenance. They claim to have

enjoyed a 52% to 65% performance enhancement in some areas.

Brzykcy, et al. [86] present an application of a Blackboard architecture to autonomous robotics which focused on updating a perception network which acts as a processing engine and storage mechanism for environmental features. It consisted of a blackboard for problem solving, processing modules and control modules. The blackboard stored a grid and vector-based maps, robot position and movement information and robots' sensor data. Data is collected from and returned to the blackboard. Each module requires no information about other modules to operate.

The use of a Blackboard Architecture for robotic learning is presented by Yang, Tian and Mei [87]. The robots query the blackboard for an action to perform and return the result back for storage in the shared database. This approach allows the robots to bypass having to determine how to perform maneuvers that have already been explored.

Fayek, Liscano and Karam [88] present work on the use of a Blackboard Architecture to control a ground robot. Sensors collect environmental data and a feature extraction module translates this data into facts that are placed on the blackboard. Based on the blackboard knowledge, user specifications, and a task decomposition routine, the robot is commanded to perform actions which impact the environment.

De Campos and de Macedo [89] present work on the use of a Blackboard-style architecture for autonomous navigation and vehicle control. A "parallel blackboard" approach, with a shared memory blackboard and area-based communications approach, was utilized. Twelve concurrent processes update and trigger off of the blackboard. The utility of a Blackboard Architecture and a geographical information system for controlling a group of UAVs in a multi-agent data integration and control system is considered by [90].

Shahbazian, Duquet and Valin [91] show how a Blackboard Architecture can be used for data fusion. They present a naval command system and a maritime surveillance system which combine data from numerous sensors to provide situational awareness.

Goldin and Chesnokov [92] present the use of a Blackboard-like architecture for spacecraft control. They divide the problem into two parts: control and information. A hierarchy is utilized for control with the system communicating with the operator and the spacecraft and communication between the information module and the spacecraft.

Deficiencies of Prior Work

The prior work presented provides a firm foundation on which to base a new system. It however, has serious deficiencies which limit system utility for planetary science purposes or in a terrestrial communications-denied environment. The Blackboard work, if it was even implemented (many papers related to this topic present theoretical and untested improvements), was generally limited by the need to have a shared memory area. Various ways of attempting to circumvent this (and the issues it created) were tried. These included asynchronous updating and triggering and the use of summary blackboards. The notion of a distributed blackboard has even been suggested.

Other work, including most of the space robotic missions, is constrained by the significant involvement of humans in the moment-to-moment control process. While this approach may be suitable for a single-large-craft mission, communications and staffing limitations are quickly reached when trying to use this approach for a multiple craft mission. Even Fink's work, which solves many of the foregoing, suffers from a single point of failure (the central control node) and numerous points of mission degradation (communications links and intermediaries). To maximize mission performance in an

environment where access is not feasible and repair is cost prohibitive, a distributed and link-loss-survivable control approach is required.

CHAPTER III

SYSTEM IMPLEMENTATION AND OPERATIONS[3]

This chapter provides an overview of a proposed multi-tier system which serves as the basis for the results, analysis and conclusions presented in subsequent chapters. It presents an algorithm for the autonomous decomposition of mission tasks, based on a controller-provided goal. This goal, which is stated as an assertion (e.g., 'a given element is present in a region' or 'enemy forces are not present along a given route') is decomposed by the autonomous control software into an initial set of sub-goals assigned to group leaders. These sub-goals may be further sub-divided and refined based on craft state and environmental conditions.

A utility-maximization, as a function of cost, metric is applied to assign follow-on tasks. The utility value is computed based upon heuristics that are utilized to estimate the value of each task that could be performed. The heuristic considers the value of previous task-type performance, the value of exploring unexplored areas and the potential that change has occurred. Cost is estimated based on historical localized movement cost and task performance estimates. This decision making process is performed at every applicable level of the hierarchy, decomposing large-scale needs into progressively smaller

[3] This chapter is derived from: Straub, J. (2011), A Modular, Application-Agnostic Distributed Control Framework for Robotic Applications. Proceedings of the International Conference on Information and Communication Technologies and Applications, Straub, J. (2013), A Data Collection Decision-Making Framework for a Multi-Tier Collaboration of Heterogeneous Orbital, Aerial and Ground Craft. Proceedings of the SPIE Defense, Security + Sensing Conference, and Straub, J. (2012), Multi-Tier Exploration Concept Demonstration Mission. Proceedings of the 2012 Global Space Exploration Conference

assignments.

Goal Definition

High-level goals are defined by mission controllers based on required mission outcomes. Analysis of the blackboard's rule set is used to determine what rules must be triggered to reach these goals. Tasking instructions are generated to trigger the rule that is determined to be the best candidate to advance the system towards triggering a final fact. Figure 1 shows high-level process used for system operations.

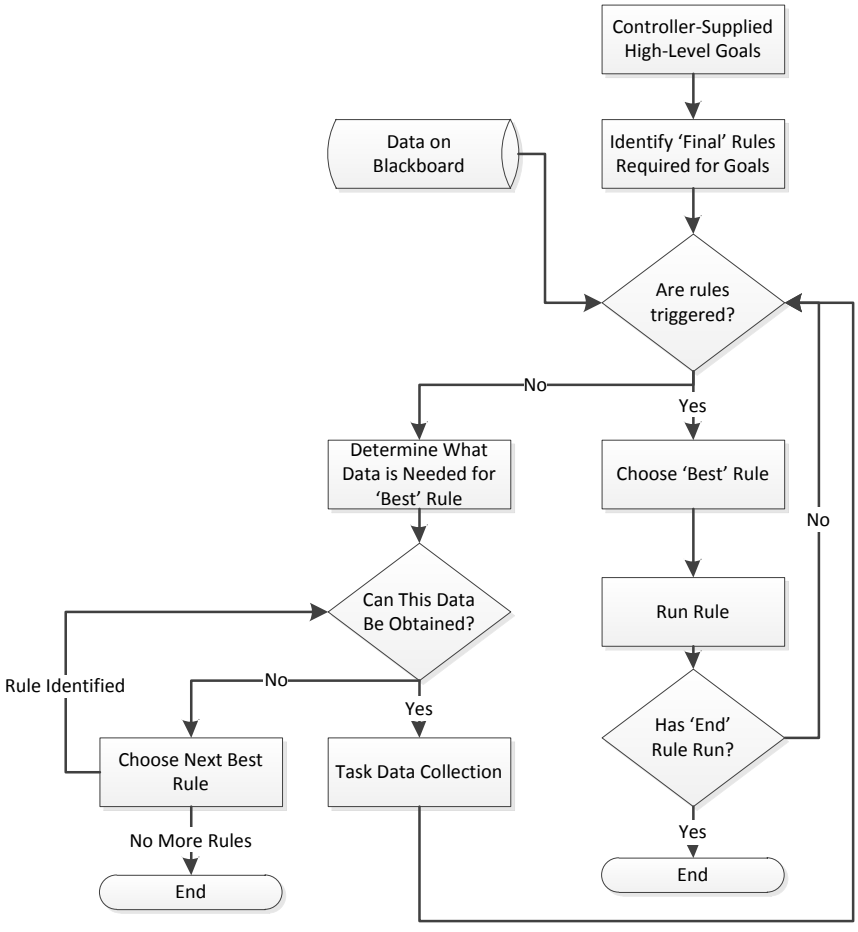

Figure 6. High-level Diagram of System Operations.

This process starts with the determination of final facts based on controller-supplied goals. Final facts are facts that, if asserted, mean that the goal has been satisfied. While multiple final facts can be identified, the assertion of any one is taken to indicate successful completion of the mission goal. Thus, the selection of multiple final facts means that there are multiple possible success conditions. If multiple facts must be triggered to indicate completion, a rule that has this combination as a precondition and asserts a final combined fact must be created.

With the final facts identified, the system begins by determining if any rules are triggered. The system will run all triggered rules before creating data collection tasks. This is based on the assumption that data collection is a comparatively expensive action; however, if some rules are similarly expensive, they can be placed into a class that require utility evaluation prior to being run.

If multiple rules are triggered, the best rule (the one that will advance the system furthest towards a final fact) is selected and run. This process iterates until either a final fact is asserted or no more rules are triggered.

If no rules are triggered, the best un-triggered rule is selected. Selection is based on a combination of three estimations: the value of triggering the rule (i.e., advancement towards final facts), the cost of data collection and the likelihood of the collected data triggering the rule. Data collection activities that satisfy multiple rules' inputs have their cost split between these rules. Figure 7 depicts the best rule determination process.

All data collection activities required to trigger the selected rule are tasked at the same time. If some required data cannot currently be collected the rule is not considered and the next-best is selected. If no rule is identified whose pre-conditions' data can be

collected, the system enters a waiting state. Once tasked data collection is complete, the system evaluates whether rules are triggered and begins the process again. Note that data collection may not trigger the identified rule if the data collected indicated a different-than-predicted fact; an alternate rule may be triggered, however.

The best rule is the one that has the highest score: the likelihood-adjusted value-units produced by the rule running divided by the cost of data collection. This process begins by computing the value of the rule running: the percentage advanced towards a final rule triggering. This percentage is a function of the number of facts required for the lowest-cost chain incorporating the rule being evaluated. For example, a chain requiring five facts of which two could be asserted by a successful run of the rule would generate a value of 40%. The projected value is determined by adjusting this based on the likelihood of data collection actually triggering the rule. This likelihood is based on the results of previous data collection and the difference between the current collection task and previous tasks.

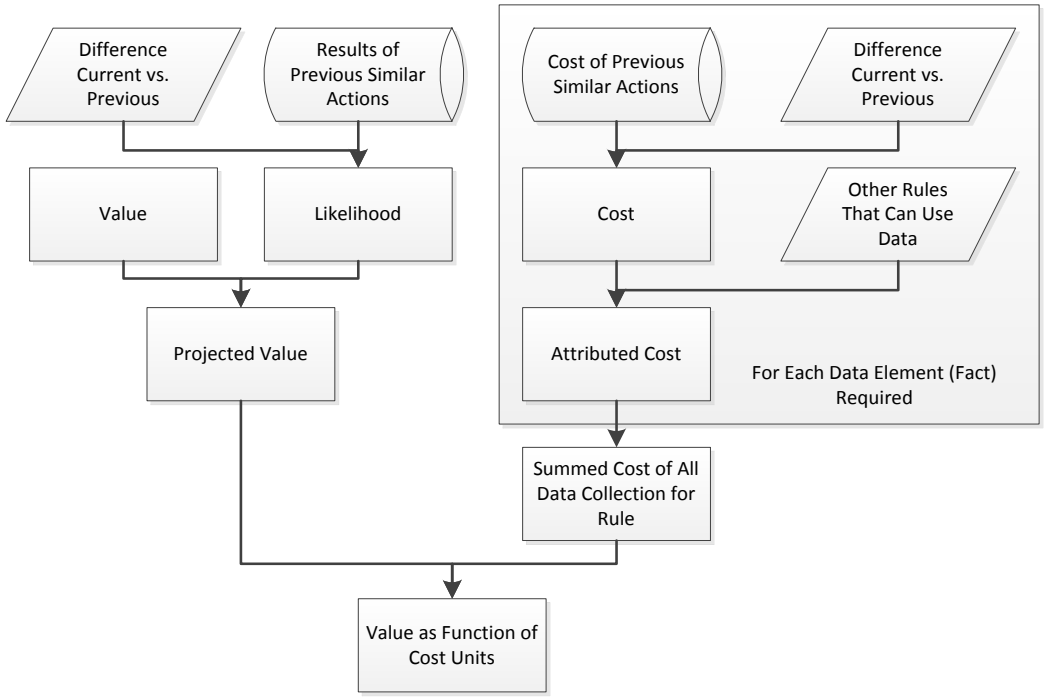

Figure 7. Depiction of the Score Determination for Each Rule.

The cost of each data collection task is determined based on the cost of similar data collection and the differences between the current and previous tasks. The attributed cost is based on dividing the cost between multiple rules to whose preconditions the data may apply. For example, if three rules could potentially use the data, one-third of the cost is attributed to each rule. The cost of all data collection required to potentially trigger the rule is summed. The score is computed by dividing the value by the cost.

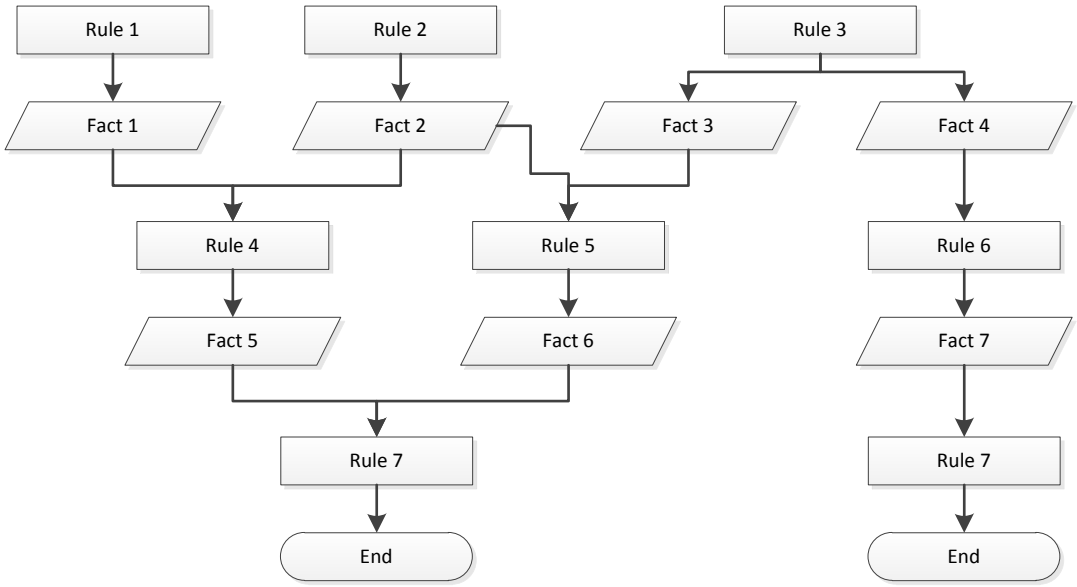

Figure 8. Rule Chain Leading to Final Rules.

Decomposition

In many cases, the execution of a chain of rules is required to cause a final rule to run. Figure 8 shows an example of a chain of rules and facts. The projected value determination approach causes rules to run in the lowest-cost path towards a final rule. Presuming that the rules had equal data collection costs, the data needed for rule 3 would be collected, as it is the first member of the lowest-cost chain (rule 3 > fact 4 > rule 6 > fact 7 > rule 7).

Data Application to Trigger Conditions

A key part of system operations is determining how to collect the data required for asserting a fact required to trigger a desired rule. Approaching the process from this direction is problematic as it requires inference without supporting data. Instead, the system assembles a catalog of collectable data and potentially assert-able facts. This database and is augmented as craft explore. For example, once a region is identified as existing, the possibility of performing appropriate types of data collection activities in the region is inserted into the database. The fact (or facts) that could be produced by each possible outcome of each prospective test is noted. For example, testing for a type of bacteria in region 5 might result in several possible outcomes: no bacteria, low-level of bacteria, medium-level of bacteria, high-level of bacteria and very-high-level of bacteria present. The produced fact may satisfy conditions requiring a particular level or conditions requiring above or below a given level.

Choosing How To Collect The Data And What Data To Collect

Multiple collection approaches can, in some cases, be used to collect the data required to assert a fact. In these cases, a collection approach must be selected. Three factors are considered: the extent to which the assertion conditions will be satisfied (and the likelihood of this occurring), ensuring that collection is balanced and comparing the utility and cost of collection.

Assertion Condition Satisfaction

Collection approaches may satisfy assertion conditions in different ways. For

example, bacteria presence may be asserted by directly testing for or observing symptoms of its presence. Both approaches could satisfy the assertion conditions; however, they may have different levels of likelihood of being successful. For example, symptoms may not be present immediately but presence may be able to be immediately detected. Alternately, the testing process for symptoms may be more robust and/or require fewer tasks. Figure 9 depicts how multiple collection approaches may be utilized to collect the data required to assert a fact.

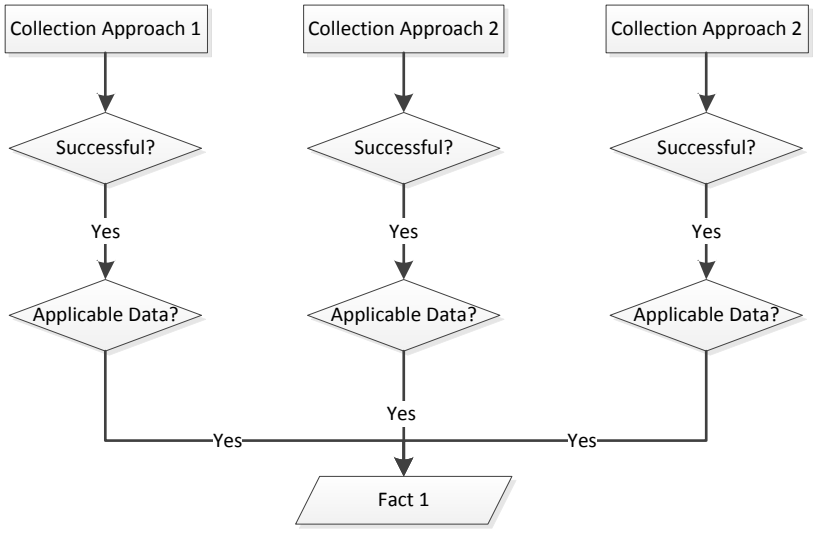

Figure 9. Multiple Collection Approaches to Assert a Fact.

Balanced Collection

Because data collection adds to the database of data available for collection and data in addition to what is specifically sought may be collected, the collection process should be balanced. It is desirable to collect data from unexplored regions and to utilize previously unused tests. Exploration benefits must be offset by the greater likelihood of greater fact assertion when utilizing known techniques and/or working in known areas.

Utility and Cost

The utility and cost of each collection approach must be compared. The utility value includes the likelihood-adjusted utility of fact assertion and the ancillary benefits produced. This is divided by the cost of collection and the method with the highest value is selected. Figure 10 depicts this process.

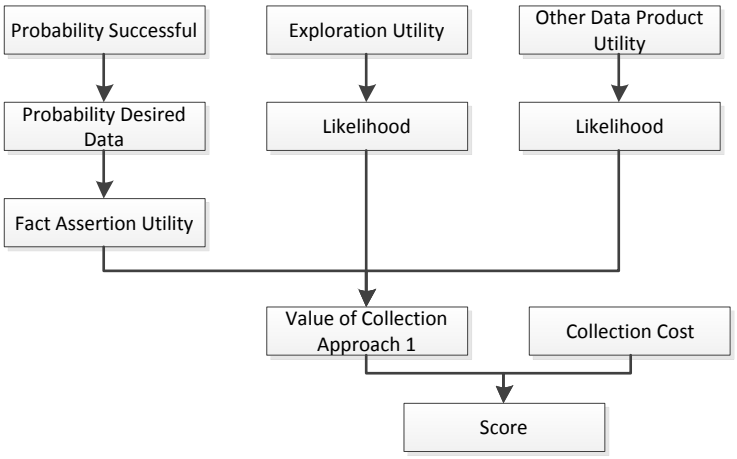

Figure 10. Data Collection Approach Score Generation Process.

Evolving Cost & Utility Heuristics

The cost and utility heuristics discussed in the previous sections are too simple for some applications. For example, different facts may have different levels of collection time and cost. Because of this, choosing a rule based upon the percentage that it moves the system down the shortest path with all facts being treated equally may be unsuitable. The selection of the shortest path may also be inaccurate because of this. Two approaches exist to solving this: a value can be assigned to each fact to characterize its relative time and resource consumption or facts could be decomposed to the point where they are approximately equivalent in terms of collection time and resource use costs. Evaluating these approaches is a subject for future work.

Delegation Across Tiers

The collection of some data may require coordinating multiple craft from multiple system tiers. For example, a UAV may be needed to assess paths for ground rovers to travel to perform data collection. In this case, a decomposed goal is assigned to a leader that further decomposes it. For this example, the UAV in a given area may be tasked with a survey and decompose this into two tasks for itself (conduct aerial survey of a given area, identify paths providing coverage of the area) and goals for three rovers. Note that in all cases the Blackboard is updated with whatever data is collected.

Methodology

Distributed command architectures have been used and proposed for various applications [16, 93-96] related to the control of multiple robots. Autonomous control is particularly needed for space exploration due to distance and delay [95]. Group autonomy is appropriate in numerous other applications. Any application where human craft-level priority-setting and control is not desirable is a candidate for group autonomy. Limited autonomy at the group level has been demonstrated [97].

Leader Node Control

The leader node is responsible for all activities of the autonomous group; however, it delegates most of this responsibility and authority and deals primarily with high-level planning, evaluation and communications with users or the higher-level tier.

For a small group, the global command module may directly control worker nodes; however, to allow larger groups, group leader modules (the AI equivalent of middle

managers) can be introduced. No leader module is expected to have an entire craft dedicated to it. The module co-exists with a worker command module on a worker craft. However, the craft selected should be one that is well suited for this use. The group leader should be easily contactable by all group members to facilitate effective management. A hierarchy of group leaders is created to manage large-scale tasks.

Each group leader's leadership control module is equivalent; it is the scope of control that differentiates them. While the overall leader communicates, accepts tasking from and reports results to system operators, lower-level leaders report results and accept tasks from their superior group leader.

Each controller is responsible for communications with other craft. At each level, the communications control system will, based on constraints, choose and schedule the most important communications for escalation to higher nodes. It also schedules contact with lower-level and peer nodes. Application-specific decision support modules assist in prioritizing application-domain information [27]. The communications control system combines craft control and data messages and queues them based on priority.

Each type of controller (command, group leader and worker) has defined modules and communications paths that can be extended for a particular application. Given this, the adaptation of a module from one application domain to another is simplified.

Worker Node Control

The control module of the worker node is responsible for local control, goal decomposition into tasks and task execution ordering. Each craft has a default task that it performs when no goals are pending. The group controller assigns the craft one or more

goals. These goals include an importance value from the analysis module of the group controller. The local controller decomposes the goals into tasks and inserts the new tasks into its task list based on a weighted combination of the task's importance, proximity and cost. The completion level of the current task is considered when determining whether to place a new task in to the first position. The planning module at the local level is prompted to reevaluate the current plan, based on the updated task list. Plan recalculation may result in the robot immediately switching to a new task.

Planning Module

The planning module at the group level is responsible for defining a strategy for completing the assigned goals. It considers constraints and assigns sub-goals, based on recommendations from the evaluation and analysis modules, to subordinates along with an assigned priority level. It also observes the progress of goal completion and re-assigns goals based on relative performance, workload and other factors.

Local planning focuses on mid-to-long-term strategies for completing assigned goals' component tasks. The module considers task location proximity and importance and the possibility of task-element concurrent performance. It also monitors completion progress, reviewing and possibly updating its plan when progress and projections differ.

Evaluation Module

The evaluation module is responsible for refining task performance estimates based on data collected during operations. The performance of all tasks conducted within the evaluation module's sphere of influence is considered and projected task completion costs

are updated based on this. The evaluation module considers the performance of particular craft relative to the group and particular task types relative to overall comparative craft performance. The outcome of this evaluation is a set of modifiers that are available to the planning module to determine costs for particular approaches to task completion. The evaluation module distributes these modifiers to all agents within its sphere of influence and to its superior controller. The evaluation module also incorporates global modifiers into its local modifier set for factors that the local group has no or limited experience with.

Evaluation at the local level focuses on the values that are used as part of the task raking process. The local evaluation module continuously refines local movement costs and costs for procedures that the craft conducts. These updated values are provided to the local group leader for incorporation in its modifier set. Modifier information from the group evaluation module is also used to update the local costing values where insufficient or out-of-date local information is available.

Analysis Module

The analysis module is responsible for problem conceptualization and solution identification. The identified solution is then developed by the planning module and executed. Analysis focuses on the identification of objects of interest (in light of mission objectives). The module is tasked with separating terrain features that are normal and uninteresting from those that are unusual or of particular mission interest (e.g., indications of water presence are of interest in Martian exploration [98]). Features of interest are assigned a priority level (corresponding to the interest level in the context of a particular objective and the objective's relative mission importance). This information is sent to the

group's planning module for incorporation in to the mission plan and subsequent assignment.

At the local level, analysis focuses on how to best complete an assigned goal. For example several sensors onboard the craft could be candidates for completing a given goal-derived task. The analysis module considers sensor capabilities in light of goal and task needs and identifies one or more sensors to use. These recommendations (note that the analysis may make multiple recommendations with associated desirability ratings to allow trade analysis) are sent to the local planning module which evaluates how to best perform the task in light of other tasks and constraints.

Executive Module

The executive module is responsible for the operations of the group. It takes requests from control system component modules and determines performance order. It is also the final arbiter of group actions and constantly checks to ensure that constraints are met, including operating requirements and craft safety constraints. Emergency response is a component of the executive module. At the group level, emergency response primarily deals with the loss of upstream contact. In this eventuality, the local group executive assumes control based on currently assigned goals and mission parameters. It also takes actions to attempt to restore upstream communications (e.g., having various subordinate crafts attempt direct communications with the group's upstream controller to rule out local interference or range issues).

At the local level the role of the executive is similar. The executive takes the plan from the planning module and turns it in to a specific set of commands that are sent to the

execution module to be further decomposed and sent to actuator controllers. The local executive also deals with emergency response, constraint checking and upstream communications failures. It overrides the planning module's plan in any instance where a constraint violation has occurred. In these instances, the executive may make an initial condition-reactive maneuver and task the planning module with refining the plan (or creating a new plan) to resolve the problematic situation.

Execution Module

The execution module is the lowest-level module and exists only as part of the worker control system. It is concerned with the physical actions that are taken by the craft (excepting communications actions controlled by the communications module). It accepts instructions from the executive module and prepares commands for transmittal to the actuator controls. It also accepts sensor input and actuator controllers' responses and transmits this information back to the executive.

Communications Module

The communications module at the group level is responsible for scheduling upstream and downstream communications based on constraints and priority. It receives inbound communications from superior and subordinate and routes them to the appropriate module for processing. It also accepts transmission requests from modules and queues and processes them. It controls local group communications by assigning certain time slots to each subordinate craft for communicating non-emergency updates. Similarly, it receives time slots that can be used for communicating updates to its superior. It will generally have

more requests than available transmission time and must use prioritization provided by the analysis module (for objective priority) and the executive to determine which requests to action (and in what order) and which to discard.

At the local level, the communications module accepts requests from local modules for communicating with the group controller and actions them based on timeslot availability and priority. It also handles requests from the group communications module to attempt to communicate with the group's upstream controller as part of a communications restoration attempt. On a group controller, group communications module tasks are performed by the local communications module. Because the local craft only communicates with its (co-located) group leader, the group communications module is the sole client of the local communications module on group controllers.

CHAPTER IV[4]

EXPERIMENTAL DESIGN AND METHODOLOGY

This chapter provides an overview of the work done to validate the multi-tier autonomous control software's performance and characterize the relative performance of the two approaches for controlling robots with heterogeneous capabilities. First, experimental goals are described. Then, system implementation is discussed. For the decentralized control approach (discussed extensively in prior chapters), an overview is provided to facilitate contrast between this approach and the centralized one. The centralized approach is described in greater detail. The experimental setup is, next, described. Finally, the testing regime utilized is presented and discussed.

Experimental Goals

Denning, et al. [99] proffer that three approaches exist to performing work in the computing sciences. The first, based on the discipline's roots in mathematics, is theoretically based and involves the use of the tools of this discipline to logically extrapolate from what is already known. The second, based on the scientific method, is predicated on the creation and validation or refutation of hypotheses. The third, based on

[4] This chapter is derived from: Straub, J. 2016. The Development of a Simulation Environment for Testing of a Multi-Tier Mission Command Architecture. Proceedings of the 2016 IEEE Aerospace Conference.

the engineering design process, views computer science as a problem-solving discipline based upon solving the needs of system users.

Denning, et al. [99], however, did not suggest that these three approaches exist or operate in a vacuum. For each of several key areas of computing, aspects relevant to each paradigm were identified. In practice, the latter two of the approaches can be synergistic. The scientific method can be useful for answering key engineering design process questions (which require empirical study) and the engineering design process can be integral in creating the experiments and experimental conditions required to perform analysis using the scientific method processes.

This work centers on this synergy, as it relates to decision making for the design of multi-craft autonomous systems. Fink [2], citing several benefits (as is typical of an engineering design process approach), has suggested that a centralized control paradigm is best suited for multi-craft control for a variety of applications. This autonomous control approach also closely mirrors the current commonly used manual control paradigm. While it is not contended that there are benefits from this methodology, it is argued that a more nuanced analysis is required to facilitate the selection of a command methodology for real-world missions.

To this end, the contribution of this work is the analysis of numerous factors that may, prospectively, impact the choice of command methodology. Each experiment utilizes the prevailing centralized control approach as a null hypothesis (H0) and then evaluates it using empirical experimentally collected data. The results are evaluated, as applicable, both in terms of statistical significance (i.e., an evaluation of whether random behavior

could have caused the difference between methodologies) and practical significance (i.e., whether the difference has any real-world importance).

Each experiment was repeated multiple times to (1) reduce the impact of any extraneous factors on the data set and (2) provide sufficient data such as to facilitate meaningful statistical significance evaluation. As is commonly known, a larger data set may facilitate the identification of smaller differences as significant (by showing that they difference recurs over numerous experiments and thus is not attributable to randomness). Thus, a higher level of repetition may have facilitated the identification of additional statistically significant findings. This, of course, could be extended ad infinitum, with each level of repetition selected yielding a suggestion that additional repetition be undertaken to see if additionally statistically significant findings might be identified. The level of repletion utilized was selected based on balancing multiple factors: the amount of time required to run some of the more computationally intensive scenarios and a desire to be able to demonstrate statistical significance for practically significant results, if applicable. A limited pre-trial experiment was performed to characterize the level of variance present in this area. This was used to determine the level of repetition that was implemented. To facilitate comparison, a single level of repetition was used across al experiments performed. In cases where data trends showed that statistical significance (at $p < 0.05$) might be attainable via additional experimentation, this is commented upon in the textual analysis. Further repetition of areas that may be of particular relevance to a various prospective applications' decision making process will serve as an area of future work.

The work presented, thus, informs the engineering design process of one that has undertaken to implement a distributed multi-craft system by facilitating the quick

comparison of the different command methodologies relative to certain mission characteristics. It also facilitates the rapid evaluation of decisions that have been made under assumed conditions as iteration in the mission planning and design process results in refinements to condition assumptions. It, thus, should facilitate a reduction in the amount of time required to make a decision as to where to further focus the design process's decision making for the command methodology.

Three goals exist for the experimentation performed. First, it seeks to characterize the performance of the Multi-Tier Autonomous Mission Architecture (MTAMA) for the control of robots with heterogeneous movement and task performance capabilities. This is performed via creating a testing environment that provides input that is relevant to potential applications for MTAMA (e.g., space exploration and persistent surveillance).

Second, it seeks to evaluate the efficacy of the MTAMA control approach for exploring an environment with limited prior knowledge (e.g., exploration of planets, moons and asteroids). It is hypothesized (H0) that the MTAMA approach will complete the characterization (a) faster and with (b) greater resource efficiency than the centralized approach.

Third, it seeks to characterize the relative performance of the centralized versus the decentralized approaches across a variety of conditions. This allows determination as to which performs best for each scenario and the extrapolation of scenario characteristics which lead each approach's superior or inferior performance. This facilitates decision making as to which approach should be used in new applications and scenarios.

System Implementation

Both systems have been implemented in C# using an object-oriented approach. Extensive reuse of the code base between the two systems has occurred to facilitate the comparison of the two approaches and minimize implementation difference impact.

A modified Blackboard approach is used in both cases, the implementation specifics (and, in particular, the differences) are highlighted in the sections that follow. In both cases, the system is based on a set of rules. Actions are initiated by rules which are triggered (by their pre-conditions being met) and executed.

Centralized Control

The centralized control approach (based conceptually on [1, 2]) places all high-level decision making in a single location (low-level decision making, such as hardware control and obstacle avoidance, is still performed onboard each craft). The approach presented herein augments Fink's concept [1, 2] with the use of elements from the Blackboard architectural approach (shown in Figure 11). The system utilizes a single centralized blackboard that resides on the orbital spacecraft and dictates the data collection needs and actions of the hierarchy of craft. An analysis of the data collection requirements for triggering rules is utilized to determine which data should be collected.

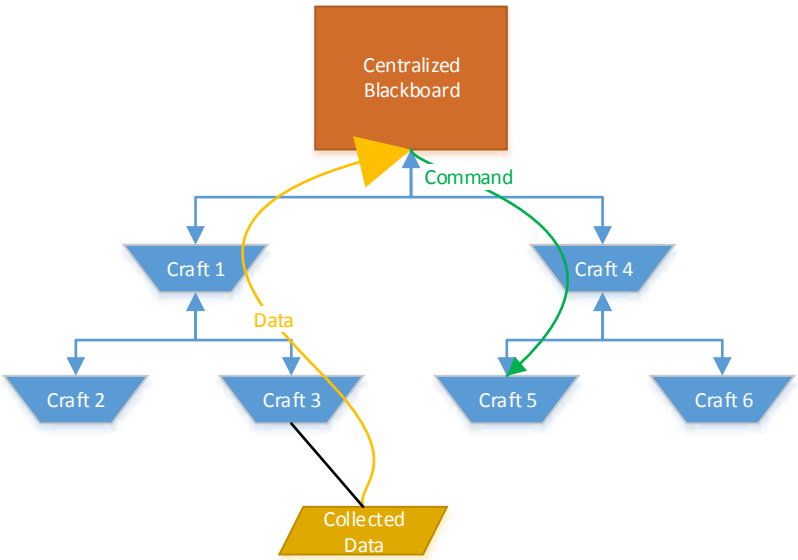

Figure 11. Centralized Control Approach.

The centralized controller devises a plan and implements a schedule that dictates what each system-member craft does. These instructions can be delivered directly from the orbital craft to the target craft or they can be relayed by intermediate craft (e.g., an aerial craft relaying to a ground craft). Individual craft perform the actions assigned to them, report task completion and send results to the orbital craft (again, this may be via another craft). Relevant assertions and data are added to the centralized blackboard. The blackboard evaluates this data and triggers and executes rules. The problem solving mechanism re-evaluates the overall plan, based on the updated state of the Blackboard, and revises goal-implementing tasks.

When changing task assignments, the centralized controller may assign one of three approaches: immediately preempt, complete current task or send report and continue. The immediately preempt instruction forces the craft to stop what it is doing and immediately begin to undertake the newly assigned task. Any relevant data is immediately sent to the central blackboard. The complete current task instruction will result in the craft completing (or trying to complete, it will still stop if the task cannot be completed, based on its initial

assignment instructions) the task at hand before moving to work on the newly assigned tasks. Finally, the report and continue approach is used if the central controller needs to know the current progress of the task (or evaluate the data collected to-date) before determining whether to preempt or wait for task completion. This, for example, would be used in a case where the central controller still considers the task at hand important (though not, now, the most important) and estimates that it is very near completion (but needs to verify this assumption through an updated status report).

Decentralized Control

This section focuses on the differences between the centralized and decentralized control approaches. It highlights critical elements of the previously described decentralized control approach which inform the experimental setup and testing regime.

The decentralized approach includes a blackboard for every craft. A principal blackboard, located on the orbital craft, contains all information relevant to achieving mission objectives. This is comprised of most of the information present on other blackboards throughout the system. Some information is abstracted on the principal blackboard, as it is important to mission objectives only when aggregated with other data (for example, an assertion may be placed on the global blackboard from the blackboard of a subordinate craft, based on data on its blackboard).

Figure 12. Distributed Control Approach.

The system operates by moving information to and from the principal blackboard and the blackboards of the subordinate craft. Each craft analyzes the information on its blackboard in terms of the rules contained on the blackboard and the goals (rules which, if triggered, constitute completion) and identifies what data to collect and/or what to delegate as goals to subordinate craft. When data collection is complete, relevant data (and assertions based on this data) are placed onto the blackboard of the craft that assigned the goal to the performing craft. Data placement may trigger a chain of actions, if rules are triggered and executed on multiple craft at levels of the mission hierarchy.

Experimental Setup

The experimental setup involves a simple simulation environment. A map with application and scenario-relevant features on it was created. This is connected to an interface layer that accepts the commands output from the control system under test and supplies the system with relevant results. The environment operates on a turn-based system to facilitate testing in faster-than-real-time. The testing environment, from the perspective of the control system under test, acts as the communications layer. In actuality, it is simulating the communications and the returned data.

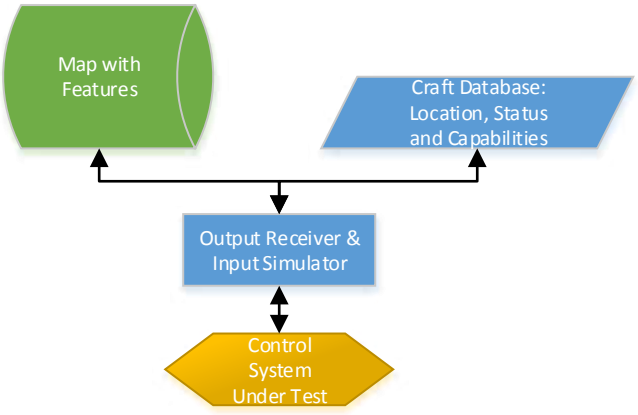

Figure 13. Testing Environment.

When the system under test sends a command to the output receiver, the command is assessed to determine what data is required from the map. This data is retrieved from the map database. Based on the configuration of the craft that the command was issued to, the terrain features in the area (and between the target and the craft's current position), other tasks assigned to the craft and other relevant details, it determines how long the task will take and supplies final and, if applicable, interim update reports at the appropriate times. Error can also be introduced at configurable levels. Error introduction is one of the experimental variables manipulated. Other elements can be introduced into the scenario,

including temporary or permanent craft incapacitation (at adjustable occurrence levels) and communications interference. The testing environment is depicted in Figure 13.

Testing Regime

The testing regime consists of six parts, each of which is now be described. First, testing was performed on each of the two systems (centralized and decentralized) to validate that they function as intended. This testing ensured that the systems being used in subsequent phases are accurate implementations of the concepts intended. Second, testing was performed to characterize the performance of both systems under basic scenarios without the addition of other factors, allowing the characterization of 'best case' performance of each of the control approaches.

Third, system performance was characterized with the introduction of data collection error. Forth, system performance was characterized with the introduction of communications issues. Fifth, system performance was characterized with the introduction of only permanent craft incapacitation. Sixth, system performance was characterized with the introduction of both temporary and permanent craft incapacitation. Seventh, system performance was characterized with the introduction of communications issues and temporary and permanent craft incapacitation. Finally, system performance was characterized with the introduction of data collection error, communications issues and temporary and permanent craft incapacitation. The level of communications errors (frequency of their occurrence and magnitude of their impact), craft incapacitation (probability of a given craft being incapacitated temporarily or permanently each turn) and data collection error (frequency of occurrence and amount of data affected) were held

constant throughout all of the eight experimental conditions, as the characterization of the systems across different levels of each affecting mechanism is a topic for future work.

CHAPTER V

A BLACKBOARD SOLVER AND PRUNING[5]

This chapter is the first of four that presents additional detail related to the system and its evaluation (previously described in Chapters III and IV). It presents a discussion of the development and testing of the blackboard solver that was integral to the operations of the Blackboard Architecture-based decision making system and the use of pruning to enhance its efficiency.

Next, an overview of the blackboard solver is provided. Then the pruning engine is discussed. Third, results and analysis related to the use of the pruning engine are presented. Finally, an overview of the results from this chapter is provided.

A Blackboard Solver

The contribution presented in this chapter is the use and characterization of a blackboard solver that implements rule, fact and/or action pruning. The blackboard solver's importance comes from the necessity of solving (determining a path through the blackboard's network of rules, facts and actions) to facilitate effective use of the Blackboard Architecture for goal-based decision making. The solver's operations begin with the identification of one or more goals to achieve. It then utilizes a routing algorithm

[5] This chapter is derived from: Straub, J. 2015. Comparing the Effect of Pruning on a Best-Path and Naïve-Approach Blackboard Solver. International Journal of Automation and Computing, Vol. 12, No. 5.

to determine what the most effective way of achieving the identified goal or goals is. A 'best path' is identified by the solver that serves as a guide for the lower-level decision making of the system robots.

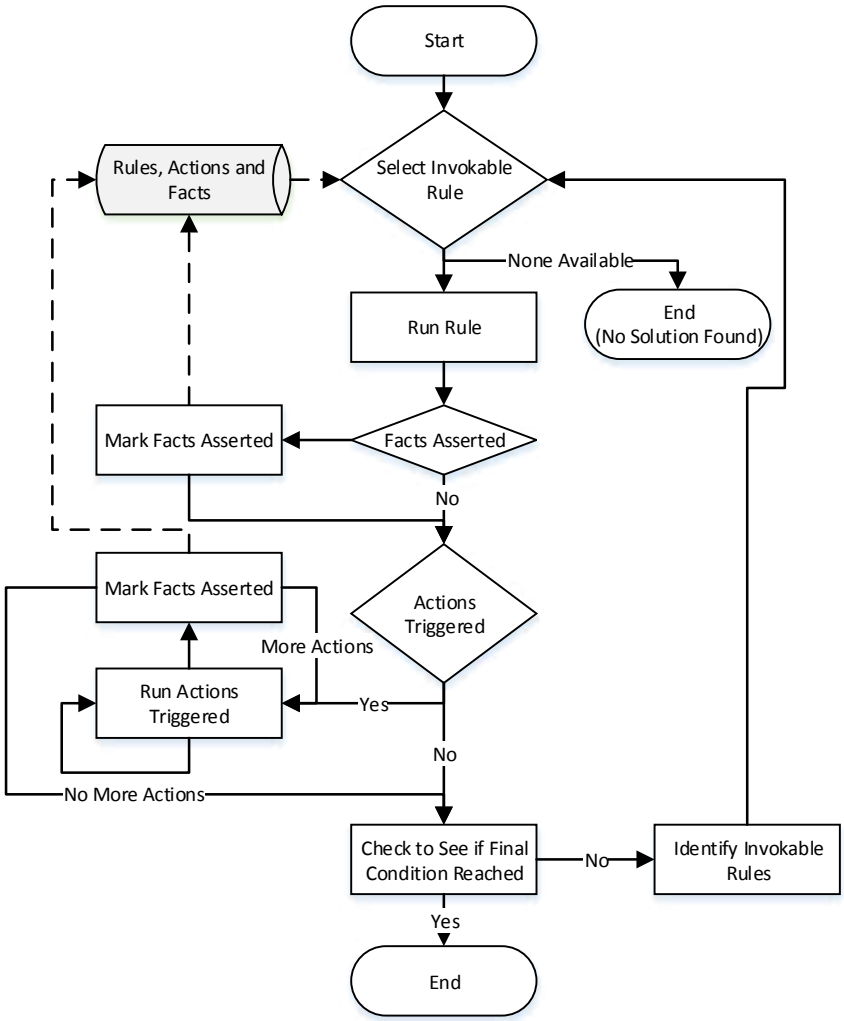

Figure 14. Naïve Solver [100].

The best path is taken to be the path that requires the lowest cost (which is a combination of the computational cost of running rules and the costs attributable to actions). In most systems that operate in a real-world environment, the action costs (e.g., the time and fuel used for moving a craft and collecting data) will dwarf the computational costs of rule activation. However, this may not always be the case. Rules requiring

particularly robust analysis may take longer than actions which do not have a physical component (e.g., triggering a message to be sent across a network). Also, the level of concurrency possible may impact this comparison as well.

The best path is identified based on predictions related to certain elements. Facts that are asserted can obviously be taken as given; however, the results of actions or rules may be unpredictable (i.e., there would be little point to collecting data which is already absolutely known; the results of data collection can be projected based on a prior knowledge and past experiences, but surprises could and should occur). Thus, for the purposes of solving for the best path, the outcomes of actions are predicted. A more complex approach (a subject for prospective future work) would be to evaluate multiple result permutations.

The naïve solver algorithm is depicted in Figure 14. It begins by selecting an invokable rule (one with all preconditions satisfied) to run (if there is not one, the algorithm ends with no solution found and the system performs its default action, typically exploration, until the blackboard's data changes or something else triggers re-solving). The rule is then run, which may or may not assert one or more facts and/or trigger one or more actions. Each action that is triggered may trigger additional actions (i.e., recursive chains of actions) and assert one or more facts. Once all facts are asserted and all actions are run, the algorithm checks to see if the designated final condition is reached. If not, the invokable rules are identified and the process restarts with the selection of an invokable rule to run.

The naïve approach is important, in its own right, for several reasons. First, the naïve approach is the typical method used by forward-only blackboard systems which look for other rules to assert once a new fact is asserted. Second, even in a solving blackboard

system (such as the one discussed) the naïve approach serves a role in dealing with dynamic data; thus, the impact of the pruning on it may be critical for systems that need to perform well during periods where an assumption is violated and an update of the blackboard network preparations for the guaranteed solver has not yet been performed. Third, there are some network configurations where the naïve solver may outperform the guaranteed one. Characterization of areas of superior naïve solver performance remains a subject for future research.

A blackboard-style system was implemented incorporating the naïve solver depicted in Figure 14 and described in the previous section. This implementation also incorporated a pruning engine, which is described subsequently and depicted in Figure 15.

Pruning Engine

The pruning engine that was developed operates iteratively. The engine begins by identifying facts that don't serve as rule conditions and facts that are not currently asserted and which cannot be asserted (e.g., there is no rule or action that asserts them). A placeholder value is then inserted into each rule which requires one of these facts as a precondition and they are removed from the list of facts to be asserted by rules and actions.

Rules that now cannot be asserted (e.g., those with the placeholder values) as well as rules with empty trigger lists are next identified and removed. Finally, actions that are no longer in any triggered list (i.e., which now cannot be invoked) are now identified and deleted. If any change was made during this iteration of the pruning engine, the process restarts (as the changes made may allow other changes to be made); if not, the engine ends.

To quantify the time required for the pruning algorithm and to test and compare the performance of the naïve solver using pruned and un-pruned data, 500 trials were run. Each trial began with the creation of a random blackboard configuration. The beginning configuration included 1,000 rules, 1,000 facts and 1,000 actions. For each fact, a random number of prerequisite facts (constrained by a maximum value parameter) was determined and this number of facts were randomly selected for use as prerequisites. For each fact and action, a random number of triggered facts and/or actions (constrained by a maximum value parameter) was determined. Whether a fact or action was used was then determined randomly for each slot. Finally, the applicable fact or action was randomly selected. A parameter-based number of facts were randomly selected to be initially asserted.

The procedure used necessarily differed for the non-pruned and pruned trials. The non-pruned trials required a two-step process. First, an alternate solver was run on the data which is guaranteed to find the best path. This was performed to allow the complexity of trials to be compared quantitatively. Second, the naïve solver was run on the blackboard. The results of the trial were recorded and the next trial commenced.

For the pruned trials, the process began by performing the pruning of the blackboard. This process continued iteratively until a run completed with no changes being made. The final number of facts, rule and actions as well as the amount of time required was recorded for each iteration. Next, the guaranteed-optimal solver was run to allow comparison of the complexity of the solution from run to run. Finally, the naïve solver was run and the results were recorded.

It is important to note that some of the networks produced may not be solvable or that the naïve solver may fail to solve networks in certain cases. The solver automatically

gives up after an amount of time that is significantly longer than the time typically required to find a solution.

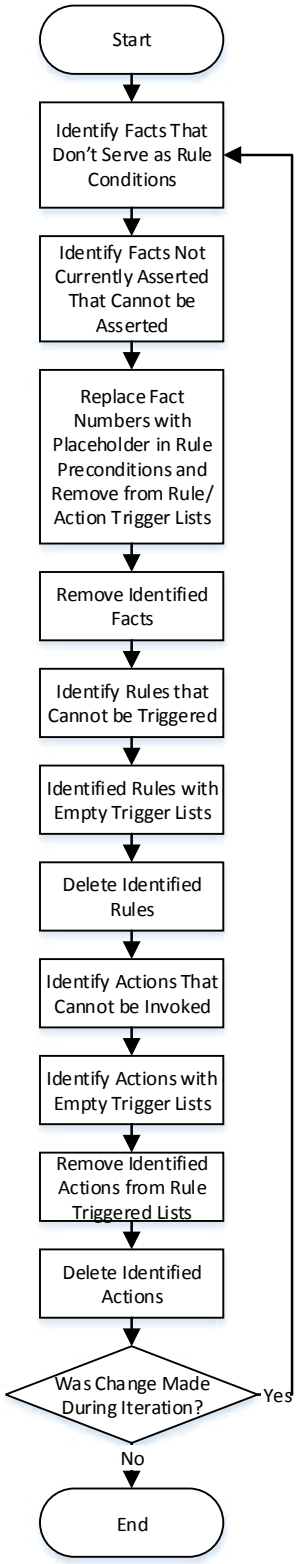

Figure 15. Pruning Engine [100].

Pruning: Results & Analysis

This section presents the data collected during the experimentation previously discussed. First, the non-pruned naïve solver results are presented in Table 1. The first four fields present the data (number of iterations, time to populate, time to solve and the path length determined) for the guaranteed-optimal solver. The remaining five fields characterize the performance of the naïve solver. The find count field indicates the number of loops of the naïve solver algorithm that were run, the rules run and acts run fields indicate the number of rules and actions invoked, respectively. The time field indicates the total time consumed by the naïve solver and the not found field indicates how many of the 500 trials resulted in no solution being identified. The time taken by the two solvers can be compared by adding the populating time and solve time for the optimal solver and comparing it to the time taken by the naïve solver. All of these time values are reported in ticks[6].

Table 1. Non-Pruned Guaranteed Optimal and Naïve Solver Results (mean values from 500 runs).

Guaranteed Optimal Solver				Naïve Solver				
# Iter	Time	Solve Time	Path Length	Find Count	Rules Run	Acts Run	Time	Not Solved
7.9	1,197.5	23.4	8.9	33.8	28,793.6	38,039.5	5,680.3	14

The data for the pruned naïve solver is divided into two tables for ease of reading. The first table (Table 2) provides the data for the pruner algorithm and the second (Table 3) provides the data for the solver. The pruner algorithm's data (in Table 2) begins with the amount of time that was required for the pruning engine to run. The next three fields

[6] Ticks [101] are the smallest unit of time measured by the Windows operating system. A tick is equal to 100 nanoseconds.

indicate the number of facts, rules and actions, respectively, which were left when the pruner completed.

In Table 3, the solver results begin with the data related to the guaranteed-optimal solver (which is located in the first four fields). The remaining five fields present the data for the naïve solver. Note that the fields in Table 3 correspond to the field in Table 1 with the same name. Thus, the description of each field will not be repeated.

Table 2. Pruned Naïve Solver Results, Pruner Time and Results (mean values from 500 runs).

Time	Facts	Rules	Actions
507,906.2	685.6	938.9	667.1

Table 3. Pruned Guaranteed Optimal and Naïve Solver Results (mean values from 500 runs).

Guaranteed Optimal Solver				Naïve Solver				
# Iter	Time	Solve Time	Path Length	Find Count	Rules Run	Acts Run	Time	Not Solved
9.6	1,317.4	20.8	11.6	14.0	12,877.8	17,747.8	2,366.0	6

The point of presenting both the guaranteed solver and naïve approaches is multi-faceted. First, it demonstrates the impact of pruning on both. The guaranteed solver's time commitment for a non-preprocessed network is actually a combination of the preparation time (i.e., the second column of Table 1 and Table 3) and the solve time (third column). This is still less than the naïve solver – across both conditions; however, it is notable that the pruning improves the naïve solver's performance significantly.

Analysis of the data presented in the previous section demonstrates the value of the pruning process to the naïve solver (a significant reduction in solver runtime). While the performance of the guaranteed-optimal approach does not change significantly (the number of iterations and path length increase slightly, as does the population time and the solve

time decreases by approximately 11%), the impact on the naïve solver is more pronounced. Comparing Table 1 and Table 3 shows that naïve solver now only requires 41.3% of the number of iterations that it did previously to generate a solution and it runs only 44.7% of the rules and 46.7% of the actions of the non-pruned approach. The number of instances where a solution could not be identified drops from 2.8% to 1.2%. Perhaps most importantly, the amount of time required decreases to 41.7% of the non-pruned approach.

The pruner, however, is computationally intensive to run, requiring an average of 507,906.2 ticks. This is, of course, much more than the average savings per solution generated (of 3,315.4 ticks). Thus, to justify the cost of the pruning, at least an average of 153.2 uses of the solver (based on dividing the amount of time required to run the pruner by the average savings per solution generation) must be run for each pruning. As the solver will typically need to be repetitively run while the blackboard system is operating (regenerating the optimal path after data on the blackboard changes), this may be a worthwhile tradeoff for many applications. The initial pruning, under the random model presented is (of course) the most expensive and, thus, even with changes to the blackboard, the benefit from the initial pruning may be enjoyed across numerous runs (with the re-pruning runs taking significantly less time due to having to do less work).

To demonstrate the lower level of cost that may be enjoyed by subsequent prunings, the amount of time required for the first three iterations of the pruner was collected across five trials. In each of these trials, the third pruner run did not produce any additional results (though this would not always be the case). This is presented in Table 4. From this, it is clear that re-prunings (which benefit from the previous prunings performed and, thus, require less work) are less expensive (requiring approximately one-half of the time of the

initial pruning).

Table 4. Comparative Cost of Pruning Iterations.

	Iteration 1	Iteration 2	Iteration 3
Max (ticks)	352841	183148	193167
Min (ticks)	293395	157702	152937
Average (ticks)	332,487	170,356.8	170,268.4
Percent	49.4%	25.3%	25.3%

Overview

This chapter has provided an overview of the research contribution of using and characterizing a blackboard solver and pruner. The solver is a key component of the creation of a goal-driven blackboard system and the pruner increases its efficacy, for some applications, and operating efficiency.

The speed enhancement provided by solving a pruned network was compared to the cost of pruning, demonstrating that approximately 153 uses of the pruned network would be required to cost-justify the pruning solely on this metric. The notion of a reducing re-pruning cost was discussed (allowing this initial cost to be spread over extended operations with a significantly lower cost level being incurred for subsequent re-prunings). However, the value of shifting time from periods of critical demand to off peak times is not considered from this purely quantitative analysis.

Pruning is an activity that can be conducted on an as-resources-are-available basis, while the benefit can be enjoyed (potentially) during times where performance is critical, such as decision making for a cyberphysical system. The comparative value of the two types of processing time consumed should, thus, also be taken into account as part of the analysis process. This relative value is (of course) application-specific and, thus, must be considered in the context of a prospective use of the Blackboard Architecture.

CHAPTER VI

SYSTEM OPERATIONS AND THE NEED FOR MAINTENANCE[7]

The previous chapter discussed pruning and demonstrated its utility, in general, for blackboard systems. The contribution of this chapter is the characterization of pruning's efficacy for the maintenance of robotic systems. This is important as, due to the nature of a Blackboard Architecture-based system for robot control, over time more and more information is added to the blackboard network and some existing or new information is or becomes irrelevant to blackboard solving. In the absence of regular maintenance to resolve this, as progressively more facts are discovered and assertions added, the speed of the system may decline. Searches will take longer, due to the amount of things to search; time-constrained searches may miss identifying critical facts or assertions, due to being forced to terminate before reaching them.

It is thus desirable to remove stale, obsolete or unused data and assertions from the blackboard and/or to archive data and assertions that, while still potentially relevant, do not appear to be likely to be used. The former can be identified by being: (a) still present after an inherent time limitation on the data, (b) supplanted by later or directly conflicting data, (c) not relevant to any rule that could be triggered (e.g., data may have been added to support a rule whose trigger condition can now never be activated due to another trigger

[7] This chapter is derived from: Straub, J. 2013. Automating Maintenance for a One-Way Transmitting Blackboard System and Other Purposes. Accepted for publication in Expert Systems.

condition being shown, through data collection, to be false) or (d) too old to be relied upon, for data that is likely to change occasionally. The latter is identified by not being relevant to any rule on the best or top-few (the exact setting can be customized as a parameter) next-best rules.

A System for Performing Ongoing Maintenance

A system for performing this ongoing Blackboard maintenance, autonomously, is now presented. The system can be activated at regular intervals. The exact interval is configured as a system parameter; however, it is expected that it will be run several time during each expiration period (the amount of time that an item on the blackboard is not rechecked for after being checked and stamped) so that only a fraction (ideally 1/3rd to 1/5th) of the blackboard items will need to be checked during each run.

Each run will assess all items on the blackboard by iterating through them. Each item on the blackboard's status will be assessed as having one of the following five statuses: current, stale / obsolete, unused, unlikely to be used, or used. The actions performed are different based on what status the item is assigned. Figure 16 presents an overview of the path taken for each possible item-status.

Current – The current status means that the item has been checked within the expiration period and does not need to be checked again at this time. When a current item is identified, no further actions are taken. The next item on the board is selected and processed.

Stale / Obsolete – Stale or obsolete items meet one of several conditions. They may be (a) data that has a definite lifetime, such as the presence of a moving robot in a particular

grid location, (b) data that has an implicit lifetime, such as the amount of an evaporating substance that remains, (c) data that changes occasionally but at unknown interval, such as weather conditions or (d) data that is replaced by different, more current data.

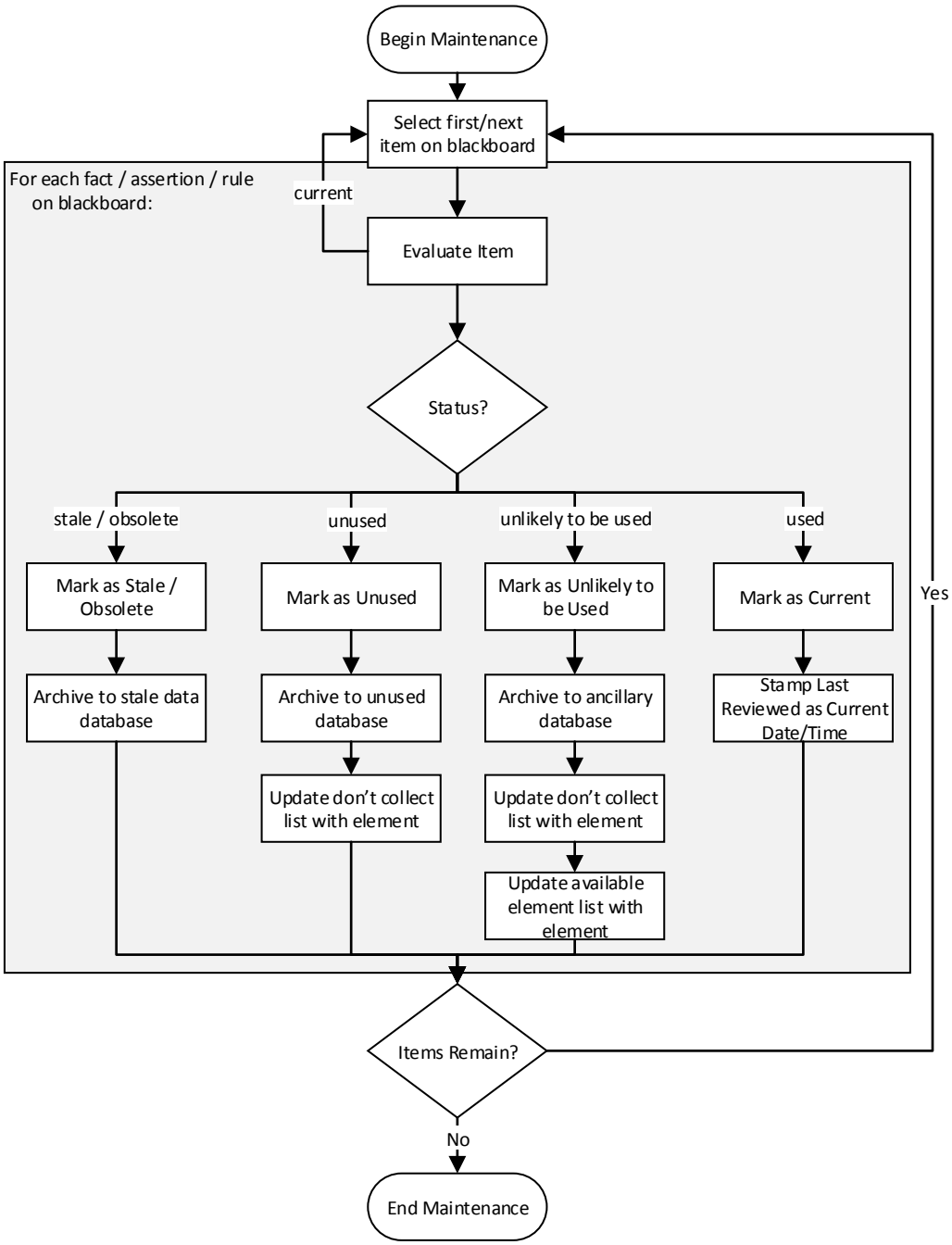

Figure 16. MTAMA Multi-Level Blackboard Architecture.

When data of each of the first three of these types is inserted into the blackboard, it is given a 'current until' expiration value. The next time the data is checked after this expiration it is removed from the blackboard. The forth is checked for during this process by looking for data with the same definition but with different timestamps. If duplicates are found, the older piece of data is marked as stale/obsolete and removed from the blackboard. Removal is effected by marking it as stale/obsolete and archiving it to the stale data database. The data can be retained in archive for a configurable period of time to facilitate system debugging (e.g., to determine why a rule executed, after the data later expires).

Unused – Unused data and assertions are data and assertions that don't meet the activation conditions for any trigger-able rule. Rules are considered able to be triggered if, for each required activation condition (or a collection of conditions meeting one triggering combination): (a) data could be collected to meet the activation conditions (e.g., it is not known that the data collection in question would return an non-applicable result), (b) another rule exists to assert the assertion that is required to trigger a give rule, (c) data already exists to meet the activation condition or (d) the required assertion has already been asserted. Thus rules become not able to be triggered if it is found that a critical data element is not as expected or a critical assertion cannot be asserted (due, for example, to the removal of another rule or the removal of an starting assertion for which there is no way to reassert).

Data and assertions that are not needed (as described above), rules that produce only unneeded assertions and rules that cannot be activated (as described above) are considered unused. When an item is determined to be unused it is marked as such and archived to the unused database. A list of collection restrictions (the 'don't collect' list) is

updated for data elements that are removed in this way to prevent effort from being wasted on trying to collect non-perishable data that will not be useful. Note that this list is checked and items removed from it if new rules are added that would make the data useful. In these instances, the data could be retrieved from the archive, if still present.

Unlikely to be Used – *i*tems are deemed unlikely to be used if they are not needed for items in the currently selected best path or one of the near-best paths. Items are deemed to be needed if they are required as part of a chain that meets an activation condition. Note that rules that end in required assertions are retained after the needed assertion has been asserted in case the assertion should be removed and be required to be re-asserted to meet the rule activation conditions. All other elements that are not needed for one of these paths and do not qualify as stale / obsolete or unused are deemed to be unlikely to be used. However, because conditions could change rendering the currently selected best and near-best paths untenable, these elements are retained in an ancillary database (items are not removed from this database, except in the case of storage limits being exceeded). Data items meeting this criteria are listed in the don't collect list to preclude effort being spent to recollect already existing data. They are also added to the available element list which is checked occasionally as part of the process of ensuring that the best and near-best paths are still actually the most desired paths and/or when best / near-best paths are rendered untenable.

Used – Elements that are used are needed by a rule that is in the currently selected best path or one of the paths identified as a near-best path. Used data is stamped with a new expiration date/time for this status (based on the expiration period) and left on the blackboard. The next item is then selected and processed.

Quantitative Analysis of Maintenance System

The data presented in the previous chapter demonstrated the computational savings afforded by using pruned data. This is, of course, offset by the cost of actually performing the data pruning required. This chapter considers the impact of different blackboard configurations on the efficacy of pruning in the context of autonomous control. It presents data from varying the initial number of rules, facts and actions as well as the number of associations between the rules, facts and actions.

First, the number of rules is varied with six different levels between 750 and 2000 rules presented. Table 5 presents the results for non-pruned operations with these rule levels. Table 6 presents the impact of pruning on operations. Then, Table 7 facilitates comparison by presenting the performance of the pruned system as a percentage of the non-pruned system.

Table 5. Non-Pruned Data for Number of Initial Rules Varied (1000 Facts, 1000 Actions, 3 Associations, times in tics).

Rules	Average Iterations	Average Time	Solve Time	Path Length	Time per Unit Length
750	5.0	739.8	6.9	5.8	1.2
1000	9.6	1731.5	17.1	12.7	1.3
1250	14.4	2974.9	40.1	25.6	1.6
1500	12.1	3272.0	23.0	15.1	1.5
1750	10.4	3102.2	18.3	11.2	1.6
2000	8.9	3283.9	17.7	9.8	1.8

Table 6. Pruned Data for Number of Initial Rules Varied (1000 Facts, 1000 Actions, 3 Associations, times in tics).

Rules	Pruning Time	#Facts	#Rules	#Actions	Average Iterations	Average Time	Solve Time	Path Length	Time per Unit Length
750	384944.1	573.6	703.5	668.3	6.5	692.9	8.9	7.6	1.2
1000	503849.3	683.6	939.1	666.2	10.2	1338.2	21.2	14.3	1.5
1250	630555.8	768.0	1173.3	668.0	12.1	2043.9	28.6	15.6	1.8
1500	763121.7	826.6	1406.6	668.5	10.4	2172.1	22.9	11.0	2.1
1750	889030.1	873.7	1640.4	666.5	9.5	2348.7	20.6	10.6	2.0
2000	1016860.7	907.1	1878.9	668.6	8.5	2420.9	18.1	8.3	2.2

Next, the number of facts is varied. Five levels of initial fact counts are used (the 1000-level is omitted as this data has already been presented in Table 5 and Table 6). Table 8 presents the non-pruned system performance, while Table 9 presents the performance of the system which utilizes pruning. Table 10, again, compares the two, presenting the performance of the pruned system as a percentage of the non-pruned system.

Table 7. Pruned Results as Percentage of Non-Pruned for Number of Initial Rules Varied (1000 Facts, 1000 Actions, 3 Associations).

Rules	Average Iterations	Average Time	Solve Time	Path Length	Time per Unit Length
750	131.0%	93.7%	129.9%	132.3%	98.2%
1000	107.1%	77.3%	123.5%	112.3%	110.0%
1250	84.2%	68.7%	71.5%	60.9%	117.4%
1500	86.4%	66.4%	99.6%	73.0%	136.4%
1750	91.5%	75.7%	112.7%	94.4%	119.4%
2000	95.2%	73.7%	101.9%	84.8%	120.1%

Table 8. Non-Pruned Data for Number of Facts Varied (1000 Rules, 1000 Actions, 3 Associations, times in tics)

Facts	Average Iterations	Average Time	Solve Time	Path Length	Time per Unit Length
750	12.4	2245.5	26.3	17.2	1.5
1250	4.9	908.9	5.5	5.1	1.1
1500	4.4	955.4	5.5	4.8	1.1
1750	3.8	820.5	4.7	4.5	1.0
2000	3.7	1368.2	7.1	4.4	1.6

Now the number of actions is varied, again using the base values of 1000 facts and 1000 rules and 3 associations. Table 11 and Table 12 present the non-pruned and pruned data, respectively. Table 13 presents a comparison between the pruned and non-pruned

systems, with the performance of the pruned system as a percentage of the non-pruned system computed.

Table 9. Pruned Data for Number of Facts Varied (1000 Rules, 1000 Actions, 3 Associations, times in tics)

Facts	Pruning Time	#Facts	#Rules	#Actions	Average Iterations	Average Time	Solve Time	Path Length	Time per Unit Length
750	404473.0	603.6	937.1	664.4	10.4	1430.0	23.2	12.6	1.8
1250	595891.3	724.9	938.2	664.2	7.0	992.9	14.9	7.8	1.9
1500	683227.3	747.1	938.8	667.7	5.5	760.3	12.2	5.8	2.1
1750	768395.4	753.2	937.9	667.1	4.5	652.8	6.2	5.2	1.2
2000	853479.1	746.5	937.9	666.8	4.3	623.2	6.4	5.0	1.3

Table 10. Pruned Results as Percentage of Non-Pruned for Number of Facts Varied (1000 Rules, 1000 Actions, 3 Associations)

Facts	Average Iterations	Average Time	Solve Time	Path Length	Time per Unit Length
750	83.4%	63.7%	88.1%	73.3%	120.3%
1250	144.5%	109.2%	271.7%	152.5%	178.1%
1500	125.3%	79.6%	220.8%	119.8%	184.2%
1750	118.2%	79.6%	132.4%	115.5%	114.6%
2000	116.5%	45.5%	89.7%	113.1%	79.3%

Table 11. Non-Pruned Data for Number of Actions Varied (1000 Rules, 1000 Facts, 3 Associations, times in tics)

Actions	Average Iterations	Average Time	Solve Time	Path Length	Time per Unit Length
750	6.4	1241.5	8.7	6.4	1.4
1250	8.8	1536.6	12.8	10.2	1.3
1500	8.1	1422.4	10.5	8.7	1.2
1750	7.2	1413.8	9.8	7.2	1.4
2000	7.6	1523.7	12.2	9.0	1.3

Table 12. Pruned Data for Number of Actions Varied (1000 Rules, 1000 Facts, 3 Associations, times in tics)

Actions	Pruning Time	#Facts	#Rules	#Actions	Average Iterations	Average Time	Solve Time	Path Length	Time per Unit Length
750	454137.7	664.3	938.4	500.7	9.5	1285.3	22.7	12.9	1.8
1250	552255.1	696.9	937.3	833.9	10.4	1362.4	22.5	13.5	1.7
1500	594598.9	709.6	937.5	999.7	10.5	1378.6	14.7	12.2	1.2
1750	650037.8	722.3	938.9	1166.6	10.5	1398.7	22.2	11.9	1.9
2000	694694.5	729.1	937.6	1336.2	11.4	1578.2	26.9	16.2	1.7

Table 13. Pruned Results as Percentage of Non-Pruned for Number of Actions Varied (1000 Rules, 1000 Facts, 3 Associations)

Actions	Average Iterations	Average Time	Solve Time	Path Length	Time per Unit Length
750	148.1%	103.5%	260.4%	200.3%	130.0%
1250	118.0%	88.7%	175.9%	131.9%	133.4%
1500	130.3%	96.9%	140.1%	140.4%	99.8%
1750	146.3%	98.9%	226.6%	164.4%	137.9%
2000	150.6%	103.6%	221.2%	179.0%	123.6%

The level of association (the number of other object types associated with each object) is now varied. Table 14 and Table 15 present association levels of 2, 4 and 5 (adding to the common association level of 3 that has been used throughout the other tables). Table 16 presents the performance of the pruned systems as a percentage of the non-pruned systems.

Table 14. Non-Pruned Data for Number of Associations Varied (1000 Rules, 1000 Facts, 1000 Actions, times in tics)

Associations	Average Iterations	Average Time	Solve Time	Path Length	Time per Unit Length
2	6.8	752.6	6.6	5.7	1.2
4	9.2	2015.7	24.1	16.1	1.5
5	12.4	3364.0	49.9	27.9	1.8

Table 15. Pruned Data for Number of Associations Varied (1000 Rules, 1000 Facts, 1000 Actions, times in tics)

Assns	Pruning Time	#Facts	#Rules	#Actions	Average Iterations	Average Time	Solve Time	Path Length	Time per Unit Length
2	272993.6	443.3	832.9	499.0	6.2	483.0	7.1	5.2	1.3
4	701340.2	817.6	969.8	749.0	13.6	2593.3	47.3	27.4	1.7
5	874828.3	898.6	981.8	798.6	12.4	2953.9	38.8	25.6	1.5

Table 16. Pruned Results as Percentage of Non-Pruned for Number of Associations Varied (1000 Rules, 1000 Facts, 1000 Actions)

Associations	Average Iterations	Average Time	Solve Time	Path Length	Time per Unit Length
2	92.0%	64.2%	107.1%	92.3%	116.1%
4	147.4%	128.7%	196.4%	170.4%	115.3%
5	99.9%	87.8%	77.7%	91.5%	85.0%

Finally, the impact of concurrently manipulating multiple variables is considered. In Table 17 and Table 18, the number of rules, facts, actions and associations is varied concurrently. Table 17 presents this data for non-pruned systems, while Table 18covers systems using pruning.

Table 19, again, presents the performance of the pruned system as a percentage of the non-pruned system.

Table 17. Non-Pruned Data for Rules, Facts and Assertions Varied Concurrently (times in tics)

Rules	Facts	Actions	Associations	Average Iterations	Average Time	Solve Time	Path Length	Time per Unit Length
750	750	750	2	6.5	516.2	6.2	5.6	1.1
1000	1000	1000	2	6.7	694.1	6.6	5.8	1.1
1250	1250	1250	4	9.4	2741.4	30.6	16.5	1.9
1500	1500	1500	4	10.1	3244.0	20.7	14.1	1.5
1750	1750	1750	5	9.3	4255.4	30.9	16.7	1.9
2000	2000	2000	5	10.2	5506.9	56.4	27.0	2.1

Table 18. Pruned Data for Rules, Facts and Assertions Varied Concurrently (times in tics)

Rules	Facts	Actions	Associations	Pruning Time	#Facts	#Rules	#Actions	Average Iterations	Average Time	Solve Time	Path Length	Time per Unit Length
750	750	750	2	152151.0	331.8	625.8	374.0	5.5	335.0	9.1	5.3	1.7
1000	1000	1000	2	271295.9	442.2	833.5	500.5	6.2	452.2	8.3	5.4	1.5
1250	1250	1250	4	1130483.6	1027.6	1212.7	938.9	12.9	2987.3	46.0	25.2	1.8
1500	1500	1500	4	1650845.5	1234.3	1455.8	1124.1	11.9	3425.7	45.5	24.8	1.8
1750	1750	1750	5	2832415.0	1572.5	1718.0	1399.1	14.2	6112.9	56.2	31.5	1.8
2000	2000	2000	5	3722875.8	1794.7	1965.1	1598.7	14.4	7354.6	76.0	38.7	2.0

Table 19. Pruned Results as Percentage of Non-Pruned for Rules, Facts and Assertions Varied Concurrently

Rules	Facts	Actions	Associations	Average Iterations	Average Time	Solve Time	Path Length	Time per Unit Length
750	750	750	2	84.7%	64.9%	147.3%	94.2%	156.4%
1000	1000	1000	2	92.7%	65.1%	124.4%	93.1%	133.6%
1250	1250	1250	4	136.6%	109.0%	150.5%	152.6%	98.6%
1500	1500	1500	4	118.4%	105.6%	219.9%	175.9%	125.0%
1750	1750	1750	5	152.9%	143.7%	181.6%	189.1%	96.1%
2000	2000	2000	5	141.2%	133.6%	134.7%	143.4%	93.9%

Network Impact on Pruning Efficacy

The results of the use of pruning at various numbers of rules, actions and facts and with different levels of associations are quite varied. The average time to prepare the network (a comparatively expensive process) is generally less with the use of pruning. This is the case with all levels of rules and 17 of 25 cases, overall.

The average path length tended to be generally more for pruned data, with 16 of the 25 cases requiring more iterations of solving for the pruned condition. A different group of 16 of the 25 cases also require a greater, on average, number of iterations, as well. In many cases, however, the differences between the two were not statistically significant at

p<0.05. In only 5 of the 20 cases did the pruning approach generate a faster average solve time; however, given the correlation between path length and solve time, this is not unexpected. The time per unit length was also, on average, higher for the pruned version; however, in many cases these differences were again not statistically significant ay p<0.05.

It is, thus, clear that the principal value of the pruning approach, generalizing across all conditions, is the reduction in the preparation time of the network (which can be two orders of magnitude greater than a single solution). This is visually depicted in Figure 17 which compares the pruned and non-pruned performance across the four experimental conditions previously discussed. Additional analysis of the variations between the conditions and between specific runs will serve as a subject for future work.

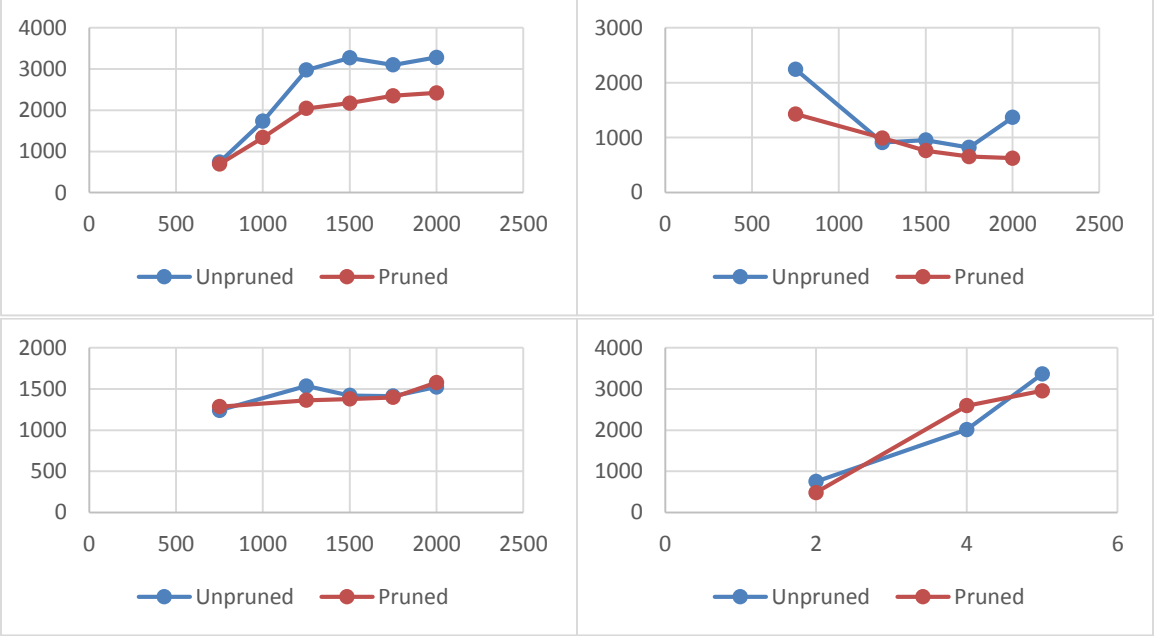

Figure 17. Comparison of pruned and unpruned performance on network preparation time for solving: varying number of rules (upper left), number of facts (upper right), number of actions (lower left) and number of associations (lower right). The X-axis represents the number of rules, facts and actions and the Y-axis represents the preparation time.

Comparing the Impact of Different Types of Pruning

Up until this point a combined pruning strategy has been considered. Under this, previously described, iterative strategy, rules, facts and actions are pruned and each pruning action may result in more objects qualifying for pruning. This section considers the impact of pruning only a subset of the object types. The results for tests, which each eliminate one type of pruning, are presented in Table 20.

Of the four, eliminating only fact pruning (condition 1) results in the best performance, in terms of several key metrics, each of which will now be discussed. It generates a significantly better (nearly 20% reduction in time) performance in terms of the average time of populating the network for solution determination. Its time per unit length is also approximately 15% lower than the base condition. Rules and actions must be traversed to determine the nature and best paths through the network; however, facts are referenced only when implicated by a rule or action. Because of this, the benefit of their reduction stems primarily from reducing the time cost of fact access (from having a smaller number to maintain and search, etc.). Pruning rules and actions, on the other hand, eliminates vestigial components of the network. This rule and action pruning provides the search benefits for the relevant object type as well as reducing the level of facts implicated (and the associated search and access costs).

Table 20. Impact of Not Pruning Certain Object Types.

Condition	Pruning Time	#Facts	#Rules	#Actions	Average Iterations	Average Time	Solve Time	Path Length	Time per Unit Length
Base (#0)	0	1000	1000	1000	8.0	1418.2	11.2	8.7	1.3
No Fact Pruning (#1)	386811.9	1000	939.2	669.7	8.2	1156.3	11.5	10.3	1.1
No Action Pruning (#2)	402558.5	702.0	1000	1000	10.0	1608.0	25.3	12.2	2.1
No Rule Pruning (#3)	387782.7	704.4	1000	664.0	9.8	1368.4	16.5	12.3	1.3

The elimination of only rule pruning (condition 3) is marginally better than the base case, with an approximately 4% decrease in network population time and a similar time per unit length. Eliminating action pruning (condition 2) – which also, largely, prevents rule pruning due to network properties – actually causes the pruning system to underperform the base (non-pruning approach), resulting in it taking 13% longer to populate the network and requiring 62% more time per unit of length.

Summary

This section has presented the research contribution of characterizing the enhancement to performance that can be provided by the pruning of a blackboard network. Specifically, it has demonstrated the value of pruning, in particular for the network preparation time, across numerous different experimental conditions (including conditions that combined multiple experimental variables). In 17 out of the 25 experimental conditions, pruning decreased network preparation time. Combinations of experimental variable also demonstrated enhanced (as compared to the base condition) performance. Combined action and rule pruning provided a 15% reduction in network preparation time, while the combination of fact and action pruning reduced preparation time by 4%. In some

cases, however, pruning was not effective at reducing network preparation time (and actually, in some cases, increased it). Combined fact and rule and fact pruning, for example, under-performed the base condition: it took 15% longer to prepare the network.

This chapter has, in addition to considering the benefits and drawbacks related to network preparation time, demonstrated that pruning has performance impacts on multiple other areas. For all of the areas of impact (network preparation and otherwise), it has characterized the areas where pruning is and is not justified, based upon the benefits provided.

CHAPTER VII

CREATING A DISTRIBUTED BLACKBOARD SYSTEM THROUGH THE USE OF BOUNDARY NODES[8]

Focus now turns to another aspect of creating a distributed blackboard system that is suitable for robotic command. The contribution of this chapter is the introduction and characterization of the use of boundary nodes to facilitate distributed blackboard operations. The proposed boundary node-based system is compared to other data synchronization and replication approaches including hierarchical, full replication, limited replication and centralized blackboard approaches.

This work was conducted in the context of the aforementioned robotic command system which is utilizes a collection of facts, rules and actions which are used to solve problems. A problem's solution (i.e., a medical diagnosis or scientific assertion) is generally determined by reaching a final fact (that represents a complete satisfaction of system requirements); however, in some cases, a system review mechanism (which characterizes the current state of the system after a period of time or an event) may be used.

Fact-rule-action chains may span the various robots of the system. This may result in a node requiring remote-to-node information to its trigger rules. New information from a given node may also be needed for decision making on other nodes. A system for

[8] This chapter is derived from: Straub, J. 2013. A Distributed Blackboard Approach Based Upon a Boundary Rule Concept. Journal of Intelligent & Robotic Systems (in press, initial online publication Sept. 30, 2015).

managing data communications between nodes is, thus, needed. Boundary nodes serve as both logical encapsulations of data as well as replication / synchronization points between the robots in the multi-robot system.

This chapter presents an analysis of the benefits and trade-offs of multiple approaches of synchronization between nodes in a distributed multi-node blackboard system. It continues with a discussion of the creation of a distributed multi-node blackboard system. Then, the use of boundary nodes for this distributed system is discussed. Next, the system is analyzed qualitatively. Following this, the quantitative data that has been collected from experimentation is presented and discussed.

Creation of a Multi-Node Blackboard System

For robotic applications, it is desirable under certain circumstances (Chapter VIII discusses when this is the case) to spread decision making across multiple robots via the use of a multi-node system. For this work, an adaption of the Blackboard Architecture is used for this purpose. To expand the blackboard/solver-based system that was discussed in the previous chapter to a multi-node system, several requirements exist. The data communication mechanism needs to be able to use low-bandwidth links effectively (without having system operations delayed by waiting for queued data transfer for extended periods of time), support peer-to-peer collaboration and interaction and facilitate the solving of problems where the data required would be on multiple nodes.

The need for low bandwidth utilization is driven by several factors, which may exist individually or in combination. Many heterogeneous craft applications will have significant bandwidth limitations between various points in the craft collection. This may

be due to craft operating at the edges of communications range, the need to transmit data products for storage, backup or additional analysis (meaning that the transmission of excess data would be constraining the ability to maximize the amount of higher-value data that was moved over a given link capacity) or link design limitations. The use of boundary nodes were identified as one prospective approach to solving this problem.

Boundary Node-Based System

Facts which are boundary objects (boundary objects are discussed in [102]) can serve to encapsulate areas of a blackboard (such as was described in [103]); alternately, they can serve to signal between different areas of the multi-node system. In the latter case, multiple boundary facts could be shared between the same two nodes to allow different types of collaboration, to facilitate the dissemination of different types of information or to solve different types of problems (or subsets of a single large problem).

Boundary facts have several characteristics:

- They are shared between the blackboards of two nodes (a multi-node boundary fact is considered as a subject for future work). Either blackboard can change the status of the fact (subject to the business logic of the system developer) and the other blackboard is notified.
- They are non-directional. Subject to the business rules of the two systems, the assertion or de-assertion of this fact can be performed by either blackboard and this will serve to fulfil (or not) the requisite input requirements for nodes which indicate the boundary fact as an input pre-condition.

- They can be final facts. This may be of little importance in some systems; however, if the system will continue operations (with a refined focus, etc.) the replication of the results of the first problem-solving process will be required.
- They can serve as inputs or outputs of rules and presumed or actual outcomes of actions on either (any) blackboard they are part of.
- They are unique and distinct within the system. Each is assigned a globally unique identifier (GUID) and this identifier is associated only with a single boundary fact.
- Multiple boundary fact links between nodes' local blackboards can be created; each can have its status modified separately.

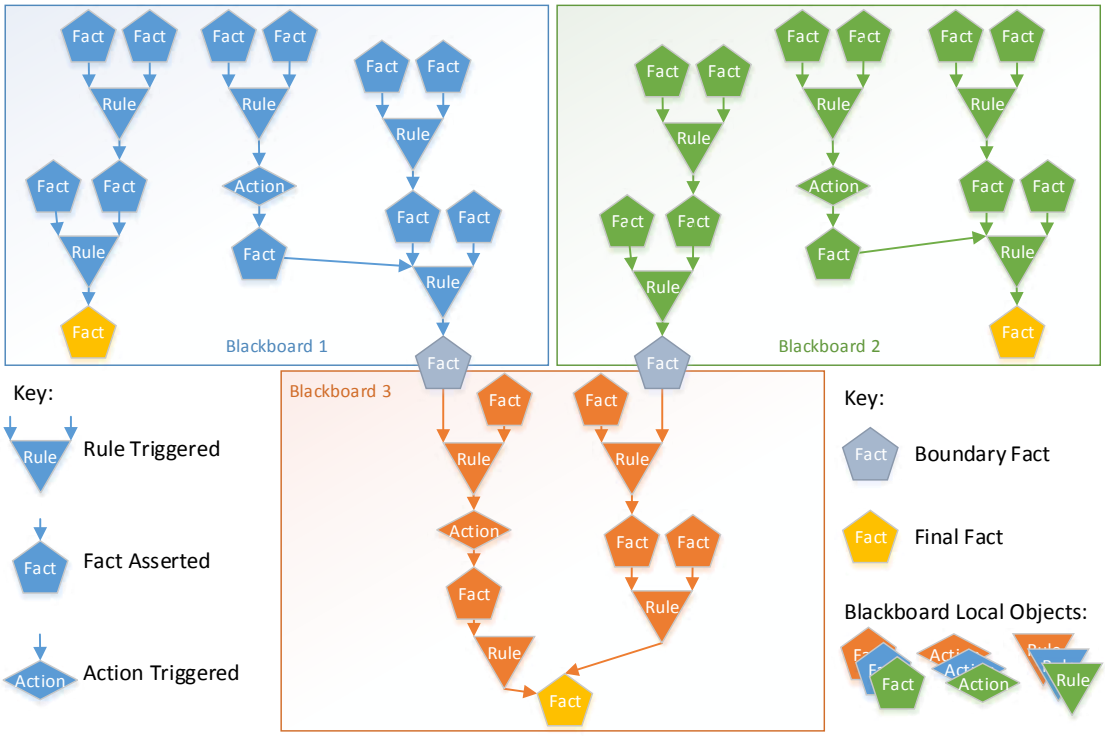

Figure 18. Blackboard spanning multiple nodes using boundary facts.

An example of a multi-blackboard system (MBS) using boundary nodes is

presented in Figure 18. Note that the links over which the status updates occur are left to the discretion of the system developer and may vary significantly from application to application.

Given the wide variety of system types that could make use of this architecture, concurrency management is largely left to the system implementation developer. For the purposes of testing, a limited locking mechanism (to prevent duplicate updates) was used. This is sufficient to allow demonstration and characterization of the concept. However, the planned robotic system will use a resultant-set-of-changes determination mechanism to facilitate system operations over an extended period with intermittent connectivity between any given set of nodes.

This work has been performed in the context of evaluating the MBS boundary node concept for use by a planned robotic system. This system will include multiple craft with heterogeneous movement, sensing and actualization capabilities.

Data Collected

To characterize the comparative performance of the proposed distributed Blackboard system, numerous multi-blackboard scenarios were created. Each scenario was randomly generated, based on the creation of a set number of agents (each with a local blackboard). The blackboards are populated randomly with a collection of facts and linking rules and actions. For the purposes of this testing, actions are presumed to always assert the projected output facts (as introducing a probabilistic model for this would serve to obscure the comparison of the distributed blackboard architectures). Each blackboard was populated with 1,000 facts and 1,000 actions and rules. Of these local facts, 400 were

initially marked as asserted. One-hundred of these facts were identified as final facts; three-hundred facts per blackboard were identified as boundary facts and a corresponding linked fact was inserted elsewhere.

The performance of the different distributed blackboard approaches (the version proposed herein, full replication, limited replication, single central blackboard and hierarchy) is characterized via running ten trials for each of four different scenarios (2 agent / blackboard, 3 agent / blackboard, 5 agent / blackboard and 10 agent / blackboard). For the hierarchy approach, arbitrary hierarchies were established; these are shown in Figure 19, Figure 20, Figure 21 and Figure 22. Each trial begins with the aforementioned randomly asserted initial facts. The agent continues running the triggered actions and rules until a final rule is reached (or a set number of iterations has completed without any final fact being reached – these non-solutions are discarded as they are not useful for comparison purposes). For each scenario, the number of iterations (each iteration consists of a single action/rule being run) and the level of replication communications activity is recorded. The amount required by other approaches, based on using the same path (rule/action order) selection is calculated. These two metrics have been selected for several reasons. Replication communications, first are selected as they are critical to understanding the impact of architecture selection on the communications requirements and system usage of the system. This is essential information for sizing a communications system (i.e., making sure that it is able to handle the magnitude and configuration of inter-craft communications needs). This is critical to inform design decisions for future work utilizing actual hardware. Second, both of these metrics are not highly application dependent, like other prospective metrics would be. This allows greater generalizability than, for example, metrics which

characterize performance in a particular mission environment or hardware configuration. This allows this work to inform future studies progressing towards multiple application areas.

The test system was custom developed. It is an enhancement of a previous (also custom) implementation [104] of a generic Blackboard architecture that has been significantly augmented to support multi-blackboard problems and, in particular, to utilize boundary objects for this purpose. The system utilizes a turn-based methodology. Prospectively, different actions can incur different time-cost levels. Physical movement times, however, were not considered for two reasons. First, they are arbitrary, and thus better left to consideration in the context of a specific set of mission objectives and circumstances. Second, they do not impact the metrics considered, with the exception of introducing an arbitrary amount of delay, which (if this is not a controlled and manipulated variable) is effectively noise being added to the data. The particular implementation for this test system was created in C#; however, this is an arbitrary selection. Replication traffic is measured by monitoring the requests made to the system to simulate data transmission / receipt.

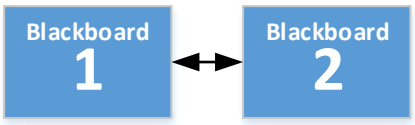

Figure 19. Diagram of two-blackboard connection.

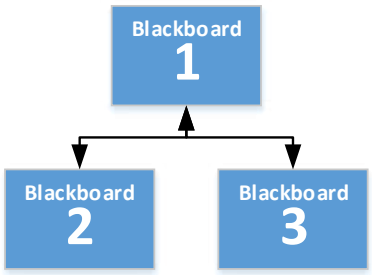

Figure 20. Diagram of three-blackboard connections.

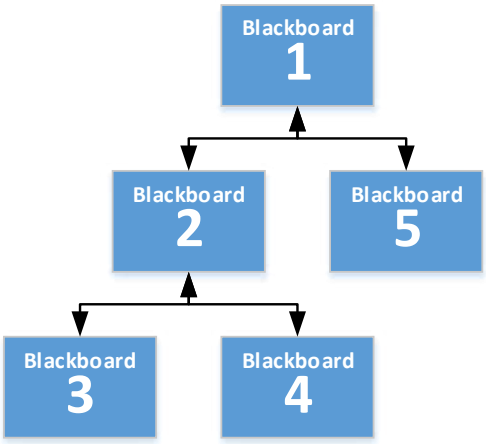

Figure 21. Diagram of five-blackboard connections.

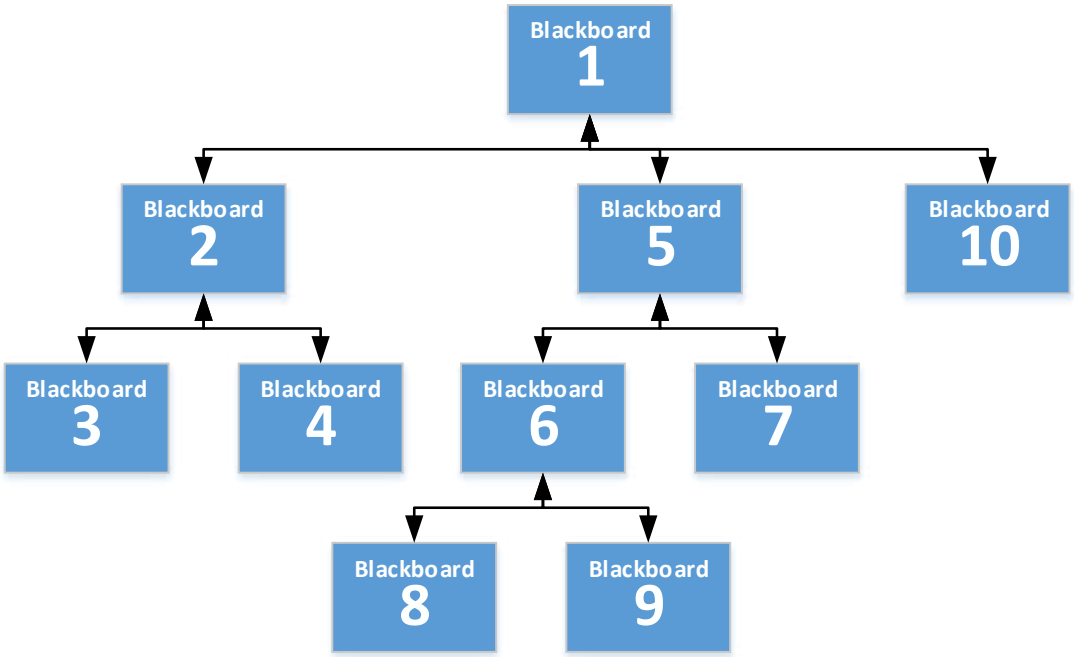

Figure 22. Diagram of ten-blackboard connections.

The results of these trials are presented in Table 21, Table 22, Table 23 and Table 24 and summarized and analyzed in the immediately succeeding section. These results correspond with the networks shown in Figure 19, Figure 20, Figure 21 and Figure 22, respectively. Table 21 presents the results for using two blackboards; Table 22 presents the results for using three blackboards. Table 23 and Table 24 present the results for using five and ten blackboards, respectively.

In addition to presenting data for the proposed approach, the tables also present several other approaches, for comparison purposes, which could also be candidates for use in a multi-robot system. A full replication approach, based on having a shared communications medium to all nodes, is presented. A limited replication approach, again requiring extensive interconnectivity, is also included. The use of a central blackboard system (where all data is sent to, and all instructions are received from a central node) is also considered. Finally, data for a hierarchical system (where the hierarchy is used to transfer / filter replication requests) is also presented.

Table 21. Results for two-blackboard testing (in terms of replication requests).

Run	Proposed	Full Replication	Limited Replication	Central Blackboard	Hierarchy
1	2	8	8	820	8
2	8	14	14	1432	14
3	0	4	4	412	4
4	2	12	12	1228	12
5	0	8	8	820	8
6	4	8	8	820	8
7	0	6	6	616	6
8	6	24	24	2452	24
9	4	12	12	1228	12
10	10	42	42	4288	42

Table 22. Results for three-blackboard testing (in terms of replication requests).

Run	Proposed	Full Replication	Limited Replication	Central Blackboard	Hierarchy
1	2	6	4	618	2
2	12	36	24	3678	19
3	3	6	6	618	4
4	11	21	22	2148	15
5	3	9	6	924	3
6	5	6	10	618	7
7	14	21	28	2148	18
8	3	6	6	618	4
9	1	3	2	312	1
10	3	3	6	312	5

Table 23. Results for five-blackboard testing (in terms of replication requests).

Run	Proposed	Full Replication	Limited Replication	Central Blackboard	Hierarchy
1	7	10	28	1030	14
2	7	10	28	1030	11
3	12	15	48	1540	21
4	11	15	44	1540	20
5	10	15	40	1540	18
6	4	5	16	520	7
7	5	10	20	1030	8
8	7	10	28	1030	13
9	14	20	56	2050	29
10	9	10	36	1030	15

Table 24. Results for ten-blackboard testing (in terms of replication requests).

Run	Proposed	Full Replication	Limited Replication	Central Blackboard	Hierarchy
1	10	10	90	1040	36
2	9	10	81	1040	25
3	18	20	162	2060	52
4	9	10	81	1040	27
5	9	10	81	1040	26
6	9	10	81	1040	28
7	26	30	234	3080	76
8	17	20	153	2060	53
9	9	10	81	1040	25
10	18	20	162	2060	49

The random placement of facts and distribution of rules and actions was selected so as to not favor any particular approach to facilitate direct comparison. An actual implementation, however, might be optimized in an application-specific manner. The limited replication approach underperformance of full replication is indicative of a non-

optimized solution. Full replication uses multicast transmissions, while limited replication utilizes point-to-point communications. Due to this, the full replication approach generally outperforms limited replication. Limited replication would, thus, generally not be used in this type of scenario (unless multicasting was impossible, in which case it would equal or outperform full replication). The hierarchy approach is based on node-to-node relaying (which is a typical feature of this approach), whereas all other approaches are point-to-point communications. It is also worth noting that the central blackboard approach presumes that the local agents must retrieve and check rules for termination (final fact assertion) conditions. If this could be performed on the central blackboard, data transfer for this approach could be reduced significantly. Whether this could be accommodated centrally or not is an implementation-specific detail.

Analysis of Data

A summary, to facilitate comparison, of the data presented in the foregoing section is included as Table 25 and visually depicted in Figure 23. From the dramatic difference in performance, it is obvious that the proposed approach significantly outperforms the limited replication, central blackboard and hierarchy approaches. It outperforms the full replication approach significantly for the two, three and five blackboard tests; however, the performance of the full replication approach is only 1.6 communications lower, on average, for the ten blackboard testing. At higher levels it appears that the full replication approach would overtake the proposed approach.

While this comparison (visually depicted in Figure 23) allows a quantitative analysis of the communications resources used by each approach, this is not the only factor

in a selection decision. The application and system configuration play a large role in this decision (in some cases larger than the performance considerations). As perhaps the most obvious example, the hierarchy approach requires a specific configuration of network connections. In the absence of this, the approach either won't work or will work virtually on top of another topology (creating a, perhaps significantly different, communications profile). Similarly, full replication requires that all nodes be connected to a single network segment. If they are not, it turns in to a (generally less efficient) hierarchy approach. The central blackboard, similarly, requires direct connectivity to the blackboard (dictating a flat network structure). Limited replication, conversely, would generally not be used on a network that can multicast (with all nodes being directly connected) as it would underperform full replication. The proposed approach, conversely, expects to have direct connectivity to any node that it shares a fact with. This allows it to exist in several different network structures. It can support peer-to-peer communications as well as communications with superior/inferior nodes. A hybrid hierarchy/proposed approach could be utilized to facilitate direct communications within the local group and use the hierarchy for transmitting to nodes outside of this group.

Table 25. Summary of averages for all testing (in terms of replication requests).

	Proposed	Full Replication	Limited Replication	Central Blackboard	Hierarchy
2 Blackboards	3.6	13.8	13.8	1411.6	13.8
3 Blackboards	5.7	11.7	11.4	1199.4	7.8
5 Blackboards	8.6	12	34.4	1234	15.6
10 Blackboards	13.4	15	120.6	1550	39.7

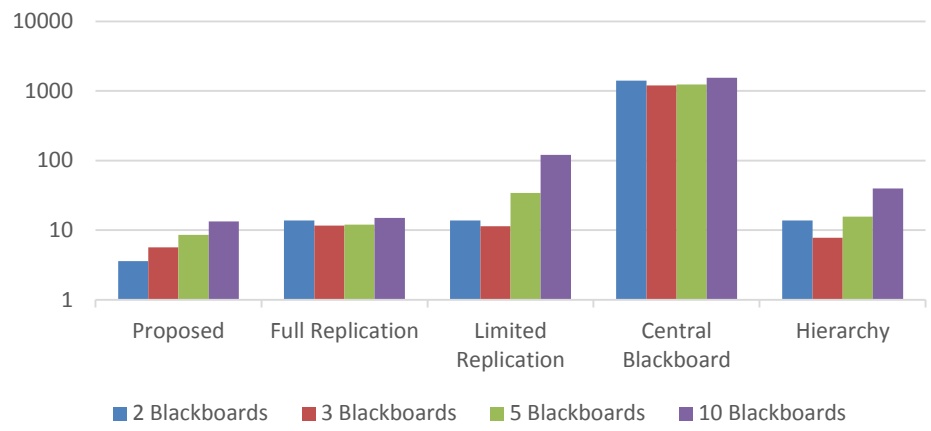

Figure 23. Comparison of Techniques (Y-axis is presented in terms of replication requests).

An example of a scenario with characteristics where the proposed architecture would excel is illustrative. One such example is a planetary exploration mission where local groups of craft conduct research in discrete areas which is designed to contribute to larger regional or planet-wide conclusion goals. These craft would have shared facts with other members of the local group which they were collaborating on specific data collection (or providing actuation in support of, etc.) and the group would have shared facts with other adjacent groups and summative shared facts which served as the relationship with higher levels in the hierarchy.

Summary

This chapter has described the research contribution of using boundary nodes as logical blackboard network and robot-to-robot intermediaries. The use of boundary nodes has been compared to other synchronization / replication approaches including hierarchical, full replication, limited replication and centralized blackboard approaches. Through this process, the efficacy of using boundary nodes was demonstrated. The full

replication approach (which relies on multicast traffic making it unsuitable for many applications), for example, required between 12% and 283% more transmission traffic than the proposed approach. Limited replication approaches require between 13.8 and 120.6 times as much traffic, while the hierarchical and central blackboard approaches require as much as 1550 and 39.7 times as many transmissions, respectively. As the foregoing demonstrates, there is significant value to using the boundary node approach, both in terms of reducing communications as well as benefiting from the associated time savings from not having to receive, process and store changes from all of the additional transactions generated by other approaches. Notably, it appears that the full (multicast-based) replication may be less expensive (in cases where multicast traffic is available) for systems with more than 10 robots / blackboards.

CHAPTER VIII

COMPARISON OF CENTRALIZED AND DISTRIBUTED COMMAND APPROACHES FOR ROBOTIC MISSIONS[9]

This chapter builds upon the work presented in prior chapters that evaluates several critical aspects of system design. It presents the contribution of comparing the efficacy and efficiency of centralized and distributed command systems under multiple experimental conditions. These conditions include both normal and impaired operations and experimental conditions that are the combination of multiple impairments. In this chapter, thus, results relevant to the key question of when centralized and decentralized command approaches are most effective are presented and analyzed.

A simulation environment was used to test the two command approaches. It utilized a 1,000 × 1,000 location grid. The premise of the test was to locate a phenomena via symptoms that are observable at different levels of data resolution, ranging from long-distance scanning to on-site analysis. For the purposes of the testing, six prospective conditions were deemed to be of interest. These conditions are part of two sets (1-3 and 4-6) with the respective positions in each set (1 and 4, 2 and 5, 3 and 6) having similar characteristics. The first and forth are observable at the lowest resolution (e.g., orbital) level, the second and fifth are observable starting at the middle (e.g., UAV) resolution level

[9] This chapter is derived from: Straub, J., R. Marsh. 2015. A Comparison of Centralized and Decentralized Blackboard Architecture-Based Command Techniques for Robotic Control Under Varying Conditions. Submitted to Expert Systems with Applications.

and the third and the sixth are observable only at the highest (e.g., ground rover) resolution level. To be an area of interest (the identification of which is the deemed completion criteria for the scenarios), a region must have a concentration of locations with both conditions three and six present. This is an analog for numerous possible mission scenarios, ranging from the identification of scientifically interesting regions to missions to locate mineral sites for extraction. There is a presumption of correlation between the presence of conditions 1, 2 and 3 and, separately, 4, 5 and 6. Thus, a location with one of the lower-resolution-detectable conditions becomes a candidate for exploration with higher-resolution equipment.

This decision making process has been embodied into the Blackboard-based architecture through the creation of an elaborate rule network comprised of over 6,000,000 facts. This network can be sub-divided, conceptually, into eight categories of facts (which are summarized in Table 26). Facts 0 to 999,999 relate to the presence of condition 1 at each of the 1,000 x 1,000 grid locations. Five more bands (facts 1,000,000 to 5,999,999) relate to conditions 2 to 5. The next 100 facts (6,000,000 to 6,000,099) relate to the suitability of regions and the last fact (6,000,100) is the final rule for the purposes of system operations (the triggering of which means that the system has successfully completed its mission). Rules and actions are denoted by their pre and post conditions and their placement in the corresponding data structure is arbitrary.

Table 26. Summary of Facts

Fact Range	Corresponds to
0 to 999,999	Condition 1 (orbital perceivable, group 1)
1,000,000 to 1,999,999	Condition 2 (aerial perceivable, group 1)
2,000,000 to 2,999,999	Condition 3 (ground perceivable, group 1)
3,000,000 to 3,999,999	Condition 4 (orbital perceivable, group 2)
4,000,000 to 4,999,999	Condition 5 (aerial perceivable, group 2)
5,000,000 to 5,999,999	Condition 6 (ground perceivable, group 2)
6,000,000 to 6,000,099	Suitability of 100 (100x100) regions
6,000,100	Final Rule (indicates mission complete)

A collection of rules and actions interconnect this network. Rules are automatically triggered, if their pre-conditions are met, and assert one or more facts. Actions are conceptually similar; however, they seek to (in this case) trigger data collection. While an outcome for each action is presumed (based on the assumption of condition correlation), this is not guaranteed. Thus, a robotic explorer (UAV or ground) may be dispatched to a location to find that the presumed outcome is not accurate. The UAV or ground robot will report any conditions that it detects at the location (or on the way, while traveling).

Figure 24 and Figure 28 depict the operations of this network control approach under successful (Figure 24) and unsuccessful (Figure 28) runs. Note that the labeling of rules and actions is arbitrary (based, for illustration purposes, on the expected result), as rules are referenced within the network by their pre- and post-conditions and actions are numbered arbitrarily (with the number being immediately stored in a corresponding rule).

Both figures exclude extraneous details. For example, given the crafts' sensing range, numerous additional facts (not relevant to the example) would be concurrently asserted (triggering corresponding rules and queueing corresponding actions). Additional facts would also be asserted, while performing actions, as the craft all sense while moving.

In Figure 24, the process starts with an orbital sensing of grid position <50,500>. Presuming (as is assumed in this example) that conditions 1 and 4 are detected, the appropriate facts are identified and asserted. Condition 1 facts are determined by multiplying the x coordinate by 1000 and adding the y coordinate so, in this case, fact 50500 is asserted to store the presence of condition 1 at this location. Condition 4 facts are determined by multiplying the x coordinate by 1000 and adding the y coordinate and 3,000,000. Thus, fact 3050500 is asserted to denote the presence of condition 4.

The identification of conditions 1 and 4 trigger (separate) processes to search for conditions 2 and 5, which are associated with (but not guaranteed by) the presence of conditions 1 and 4, respectively, at a location. This is done by triggering an action that will assign a nearby air-based craft to explore this region. If conditions 2 and 5 are detected (supporting the possibility of conditions 3 and 6 being present), than a similar process will occur. Facts 1050500 and 4050500 will be asserted to store the presence of these conditions and actions will be triggered to explore this grid area with a ground-based craft.

If the ground based craft confirms the presence of both conditions 3 and 6, this will identify the grid location as a target location and support the triggering (along with the presence of other targets) of final rule 6000100, when a sufficient number of target locations have been identified in the region.

A portion of this process, from actual operations, is shown in Listing 1 which uses the data sensed from the map shown in Figure 25 (a detail view of the top-left 200x200 grid locations is also shown, for ease of viewing, in Figure 27 and a key to the coloration is shown in Figure 26). Three elements are highlighted to illustrate key portions of the process.

The collection of data by robotic exploration, for condition 1, is highlighted in yellow in Listing 1. Fact 1101461 is asserted, triggering rule (F1101461) >> (A507306), which launches action 507306. The same process, for condition 2, is highlighted in red. Fact 3105461 is asserted, triggering rule (F3105461) >> (A527307) and launching action 527307. From the numbering of the facts (and the discussion of the fact numbering system, previously), it is clear that these two facts relate to the same grid coordinate.

The assertion of the final rule is also highlighted (in magenta). This indicates that the assertion of facts 2922586 and 5922586 (related to a different grid coordinate than the previous example) causes rule (F2922586, F5922586) >> (F6000100) to run. Shortly thereafter, the system again checks to see if final rule 6000100 has been asserted and, when it does, it determines that the mission has been completed and stops.

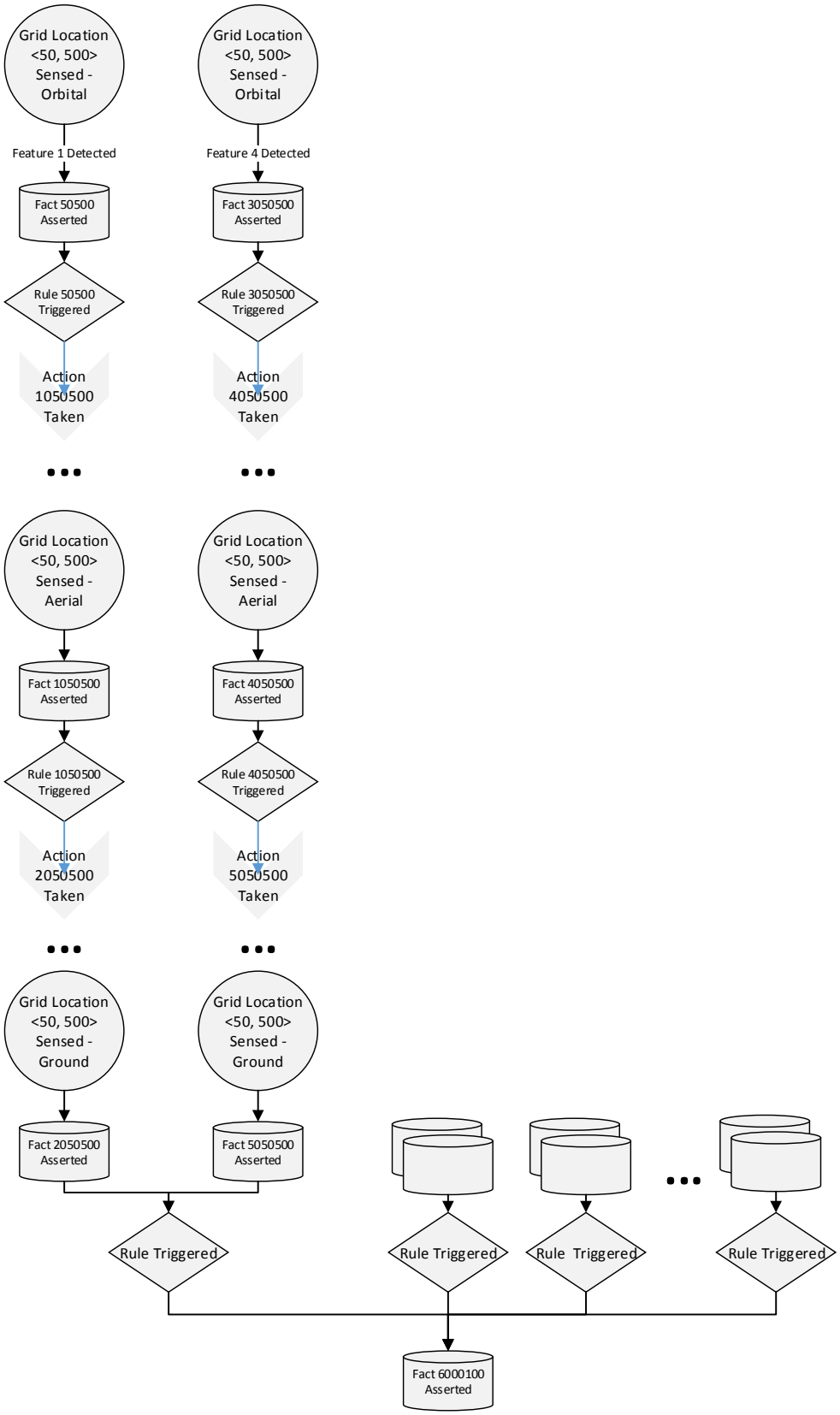

Figure 24. Ideal operations of the Blackboard-based control network.

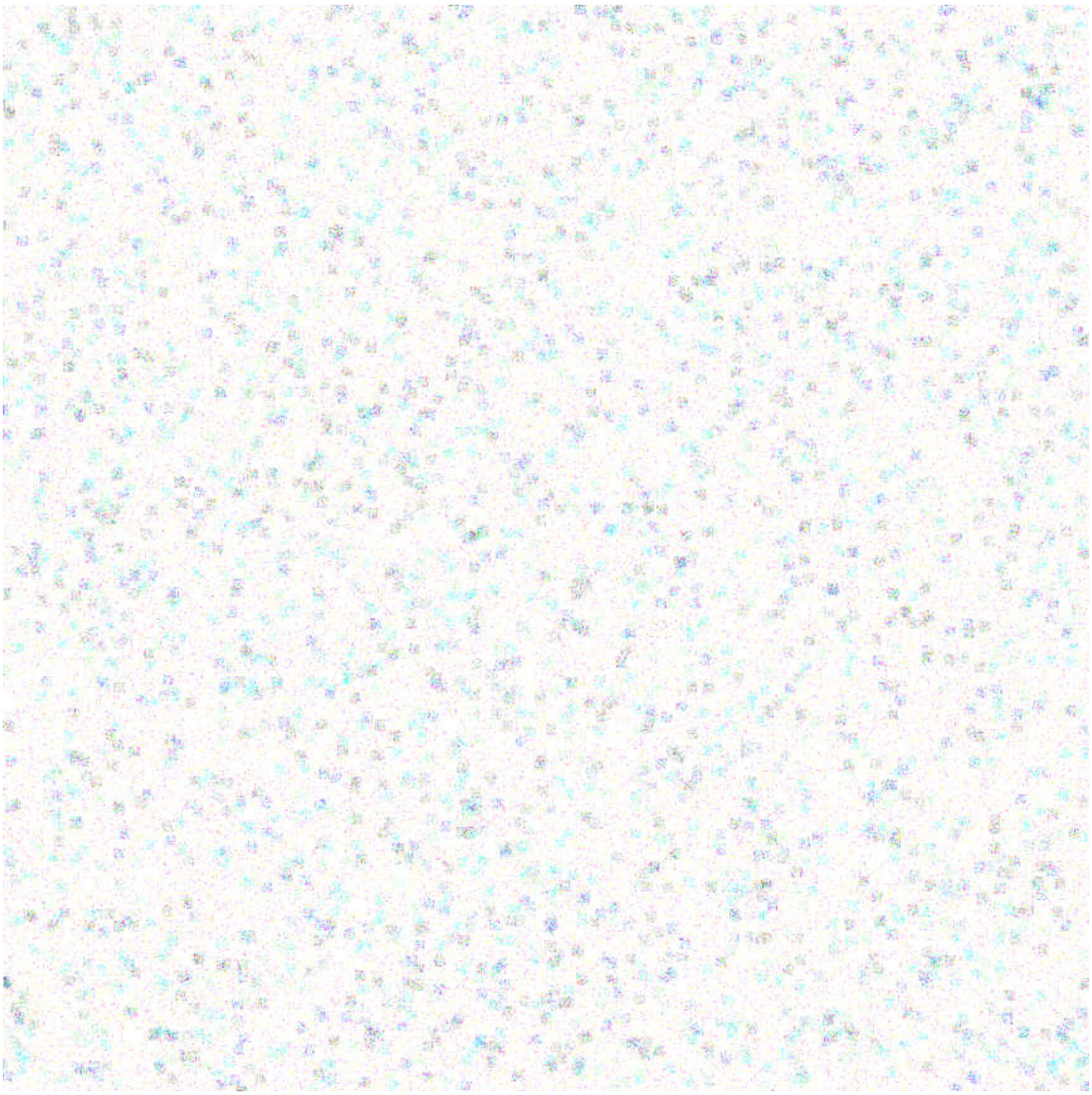

Figure 25. Global map for example (coloration key can be found in Figure 26).

Under this ideal scenario, the system could theoretically operate in forward-only mode and be successful. The process becomes more complex when non-ideal locations (such as shown in Figure 28) are present. In Figure 28, the presumption of the presence of condition 5 is not accurate. Thus, when an aerial sensing of this location occurs, fact 4050500 is not asserted. This prevents the remainder of the network from triggering.

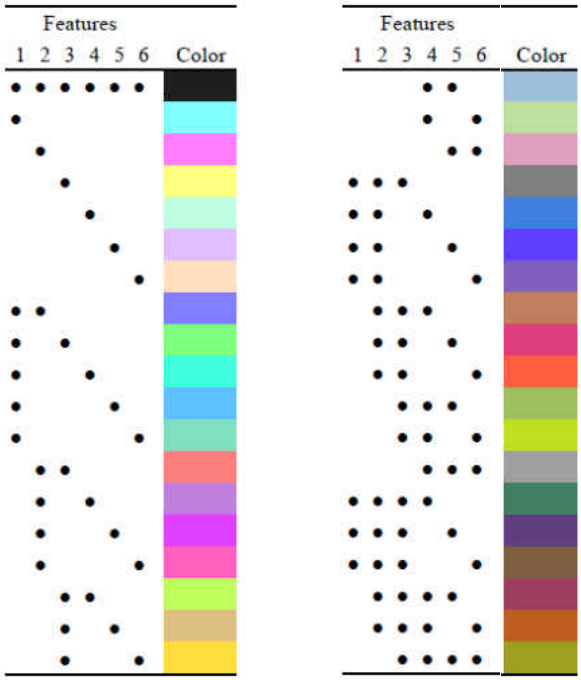

Figure 26. Map Key.

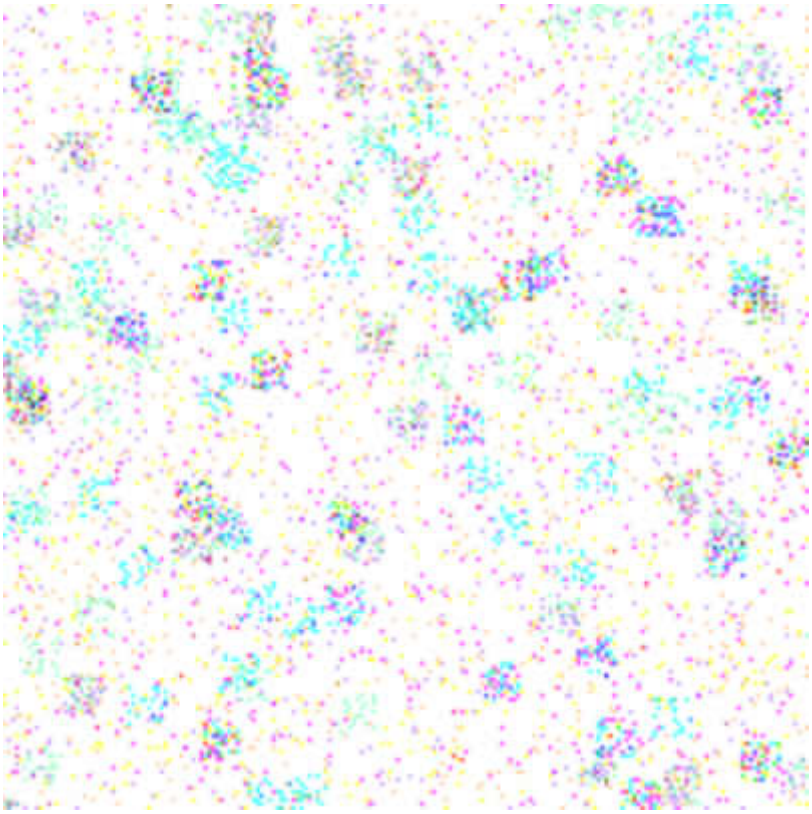

Figure 27. Top-left 200 x 200 grid locations for example (coloration key can be found in Figure 26).

Listing 1. Sample Log of Operations

FACT 4152504 Asserted	RULE (F4238990) >> (A1194953) has run
FACT 1154504 Asserted	ACTION 1194953 triggered
FACT 4897858 Asserted	RULE (F509796) >> (A2548980) has run
FACT 1898858 Asserted	ACTION 2548980 triggered
FACT 5834010 Asserted	RULE (F511796) >> (A2558980) has run
FACT 3236990 Asserted	ACTION 2558980 triggered
FACT 4238990 Asserted	RULE (F3530148) >> (A2650742) has run
FACT 5239990 Asserted	ACTION 2650742 triggered
FACT 973805 Asserted	RULE (F3531148) >> (A2655742) has run
FACT 974805 Asserted	ACTION 2655742 triggered
FACT 1976805 Asserted	RULE (F3544983) >> (A2724917) has run
FACT 218314 Asserted	ACTION 2724917 triggered
FACT 3530148 Asserted	RULE (F3547983) >> (A2739917) has run
FACT 3531148 Asserted	ACTION 2739917 triggered
FACT 1884008 Asserted	RULE (F1884008) >> (A4420041) has run
FACT 2893278 Asserted	ACTION 4420041 triggered
FACT 4894278 Asserted	RULE (F4894278) >> (A4471393) has run
FACT 3544983 Asserted	ACTION 4471393 triggered
FACT 3547983 Asserted	RULE (F4897858) >> (A4489293) has run
FACT 1101461 Asserted	ACTION 4489293 triggered
FACT 2103461 Asserted	RULE (F1898858) >> (A4494291) has run
FACT 3105461 Asserted	RULE (F509796) >> (A2548980) has run
FACT 4151203 Asserted	RULE (F511796) >> (A2558980) has run
FACT 175138 Asserted	RULE (F3530148) >> (A2650742) has run
FACT 1176138 Asserted	RULE (F3531148) >> (A2655742) has run
FACT 177138 Asserted	RULE (F3544983) >> (A2724917) has run
FACT 3177138 Asserted	RULE (F3547983) >> (A2739917) has run
FACT 509796 Asserted	RULE (F1884008) >> (A4420041) has run
FACT 511796 Asserted	RULE (F4894278) >> (A4471393) has run
FACT 5147835 Asserted	RULE (F4897858) >> (A4489293) has run
FACT 3150835 Asserted	RULE (F1898858) >> (A4494291) has run
RULE (F1101461) >> (A507306) has run	RULE (F973805) >> (A4869025) has run
ACTION 507306 triggered	RULE (F974805) >> (A4874025) has run
RULE (F3105461) >> (A527307) has run	RULE (F1976805) >> (A4884026) has run
ACTION 527307 triggered	FACT 2237988 Asserted
RULE (F3150835) >> (A754177) has run	FACT 5237988 Asserted
ACTION 754177 triggered	FACT 4239988 Asserted
RULE (F4151203) >> (A756018) has run	FACT 4240988 Asserted
ACTION 756018 triggered	FACT 4241988 Asserted
RULE (F4152504) >> (A762523) has run	FACT 973805 Asserted
ACTION 762523 triggered	FACT 974805 Asserted
RULE (F1154504) >> (A772521) has run	FACT 1976805 Asserted
ACTION 772521 triggered	FACT 5532147 Asserted
RULE (F175138) >> (A875690) has run	FACT 1884008 Asserted
ACTION 875690 triggered	FACT 4544984 Asserted
RULE (F1176138) >> (A880691) has run	FACT 3546984 Asserted
ACTION 880691 triggered	FACT 4923482 Asserted
RULE (F177138) >> (A885690) has run	FACT 1118159 Asserted
ACTION 885690 triggered	FACT 5409249 Asserted
RULE (F3177138) >> (A885692) has run	FACT 101459 Asserted
ACTION 885692 triggered	FACT 2103459 Asserted
RULE (F218314) >> (A1091570) has run	FACT 2176139 Asserted
ACTION 1091570 triggered	FACT 507797 Asserted
RULE (F3236990) >> (A1184952) has run	FACT 4148836 Asserted
ACTION 1184952 triggered	FACT 2237988 Asserted

FACT 5237988 Asserted FACT 4239988 Asserted	FACT 4240988 Asserted FACT 4241988 Asserted

• • •

RULE (F922535) >> (A4612675) has run RULE (F1922535) >> (A4612676) has run RULE (F1922537) >> (A4612686) has run RULE (F4922545) >> (A4612728) has run RULE (F4922548) >> (A4612743) has run RULE (F4922580) >> (A4612903) has run RULE (F922585) >> (A4612925) has run RULE (F922586) >> (A4612930) has run RULE (F2922586, F5922586) >> (F6000100) has run RULE (F922587) >> (A4612935) has run RULE (F4922587) >> (A4612938) has run RULE (F922589) >> (A4612945) has run RULE (F922591) >> (A4612955) has run RULE (F1922598) >> (A4612991) has run RULE (F1922666) >> (A4613331) has run RULE (F922670) >> (A4613350) has run RULE (F922673) >> (A4613365) has run RULE (F4923482) >> (A4617413) has run RULE (F4923518) >> (A4617593) has run RULE (F4923519) >> (A4617598) has run RULE (F3923523) >> (A4617617) has run RULE (F3923525) >> (A4617627) has run RULE (F3923526) >> (A4617632) has run RULE (F923534) >> (A4617670) has run RULE (F923537) >> (A4617685) has run RULE (F923538) >> (A4617690) has run RULE (F1923550) >> (A4617751) has run RULE (F1923572) >> (A4617861) has run RULE (F1923578) >> (A4617891) has run RULE (F923589) >> (A4617945) has run RULE (F923590) >> (A4617950) has run RULE (F1923629) >> (A4618146) has run RULE (F4923643) >> (A4618218) has run RULE (F4923645) >> (A4618228) has run RULE (F923669) >> (A4618345) has run RULE (F4923699) >> (A4618498) has run RULE (F1923716) >> (A4618581) has run RULE (F1923726) >> (A4618631) has run RULE (F1923741) >> (A4618706) has run RULE (F4924484) >> (A4622423) has run RULE (F1924489) >> (A4622446) has run RULE (F4924492) >> (A4622463) has run RULE (F1924493) >> (A4622466) has run RULE (F1924499) >> (A4622496) has run RULE (F4924504) >> (A4622523) has run RULE (F1924507) >> (A4622536) has run RULE (F3924508) >> (A4622542) has run RULE (F3924509) >> (A4622547) has run RULE (F4924510) >> (A4622553) has run	RULE (F4924522) >> (A4622613) has run RULE (F1924529) >> (A4622646) has run RULE (F924530) >> (A4622650) has run RULE (F1924532) >> (A4622661) has run RULE (F924534) >> (A4622670) has run RULE (F4924534) >> (A4622673) has run RULE (F924538) >> (A4622690) has run RULE (F1924538) >> (A4622691) has run RULE (F4924550) >> (A4622753) has run RULE (F1924557) >> (A4622786) has run RULE (F1924560) >> (A4622801) has run RULE (F1924580) >> (A4622901) has run RULE (F924587) >> (A4622935) has run RULE (F924589) >> (A4622945) has run RULE (F4924655) >> (A4623278) has run RULE (F924671) >> (A4623355) has run RULE (F924672) >> (A4623360) has run RULE (F4924707) >> (A4623538) has run RULE (F4924714) >> (A4623573) has run RULE (F1924741) >> (A4623706) has run RULE (F1927759) >> (A4638796) has run RULE (F4935764) >> (A4678823) has run RULE (F4938771) >> (A4693858) has run RULE (F1940769) >> (A4703846) has run RULE (F4940771) >> (A4703858) has run RULE (F3941772) >> (A4708862) has run RULE (F3941774) >> (A4708872) has run RULE (F3942772) >> (A4713862) has run RULE (F4942772) >> (A4713863) has run RULE (F3942774) >> (A4713872) has run RULE (F4943774) >> (A4718873) has run RULE (F3943775) >> (A4718877) has run RULE (F3944776) >> (A4723882) has run RULE (F3945774) >> (A4728872) has run RULE (F3945775) >> (A4728877) has run RULE (F4945776) >> (A4728883) has run RULE (F3946775) >> (A4733877) has run RULE (F4946778) >> (A4733893) has run RULE (F947005) >> (A4735025) has run RULE (F4947777) >> (A4738888) has run RULE (F4948778) >> (A4743893) has run RULE (F1948781) >> (A4743906) has run RULE (F1952783) >> (A4763916) has run RULE (F4959792) >> (A4798963) has run RULE (F4954787) >> (A4773938) has run RULE (F4956789) >> (A4783948) has run RULE (F3957788) >> (A4788942) has run RULE (F4957789) >> (A4788948) has run RULE (F4959788) >> (A4798943) has run RULE (F3959790) >> (A4798952) has run

RULE (F3959791) >> (A4798957) has run	RULE (F966795) >> (A4833975) has run
RULE (F4959791) >> (A4798958) has run	RULE (F966796) >> (A4833980) has run
RULE (F3959792) >> (A4798962) has run	RULE (F966797) >> (A4833985) has run
RULE (F3960792) >> (A4803962) has run	RULE (F967797) >> (A4838985) has run
RULE (F3960793) >> (A4803967) has run	RULE (F1967797) >> (A4838986) has run
RULE (F4960793) >> (A4803968) has run	RULE (F967800) >> (A4839000) has run
RULE (F3961790) >> (A4808952) has run	RULE (F968797) >> (A4843985) has run
RULE (F961793) >> (A4808965) has run	RULE (F1968800) >> (A4844001) has run
RULE (F3961793) >> (A4808967) has run	RULE (F1969798) >> (A4848991) has run
RULE (F4961793) >> (A4808968) has run	RULE (F970800) >> (A4854000) has run
RULE (F4962792) >> (A4813963) has run	RULE (F1971801) >> (A4859006) has run
RULE (F3962793) >> (A4813967) has run	RULE (F1971804) >> (A4859021) has run
RULE (F1963792) >> (A4818961) has run	RULE (F1972802) >> (A4864011) has run
RULE (F4963792) >> (A4818963) has run	RULE (F973804) >> (A4869020) has run
RULE (F4963794) >> (A4818973) has run	RULE (F1973804) >> (A4869021) has run
RULE (F1963795) >> (A4818976) has run	RULE (F973805) >> (A4869025) has run
RULE (F3964794) >> (A4823972) has run	RULE (F974805) >> (A4874025) has run
RULE (F4965794) >> (A4828973) has run	RULE (F1976805) >> (A4884026) has run
RULE (F1965795) >> (A4828976) has run	RULE (F980005) >> (A4900025) has run
RULE (F3965797) >> (A4828987) has run	FACT 6000100 Asserted - Mission Accomplished

This occurrence illustrates the need for the solver mechanism that has been previously described. Because the solver always works backwards from the goal (in this case final fact 6000100), any rule, fact or action that is not in a chain to this will not be identified as a goal and thus no effort will be made to further explore areas that have no pathway to achieving the overall system goals. For example, if the non-presence of condition 5 was known at the time, no ground robot tasking would be performed to seek condition 3 as, even if condition 3 was detected, this would not advance the system towards its goal.

Some networks will be inherently unsolvable due to a failure to have enough (or any) target locations. Other networks may be unsolvable under conditions that impair the ability of the system to fully function. This is considered in the subsequent section.

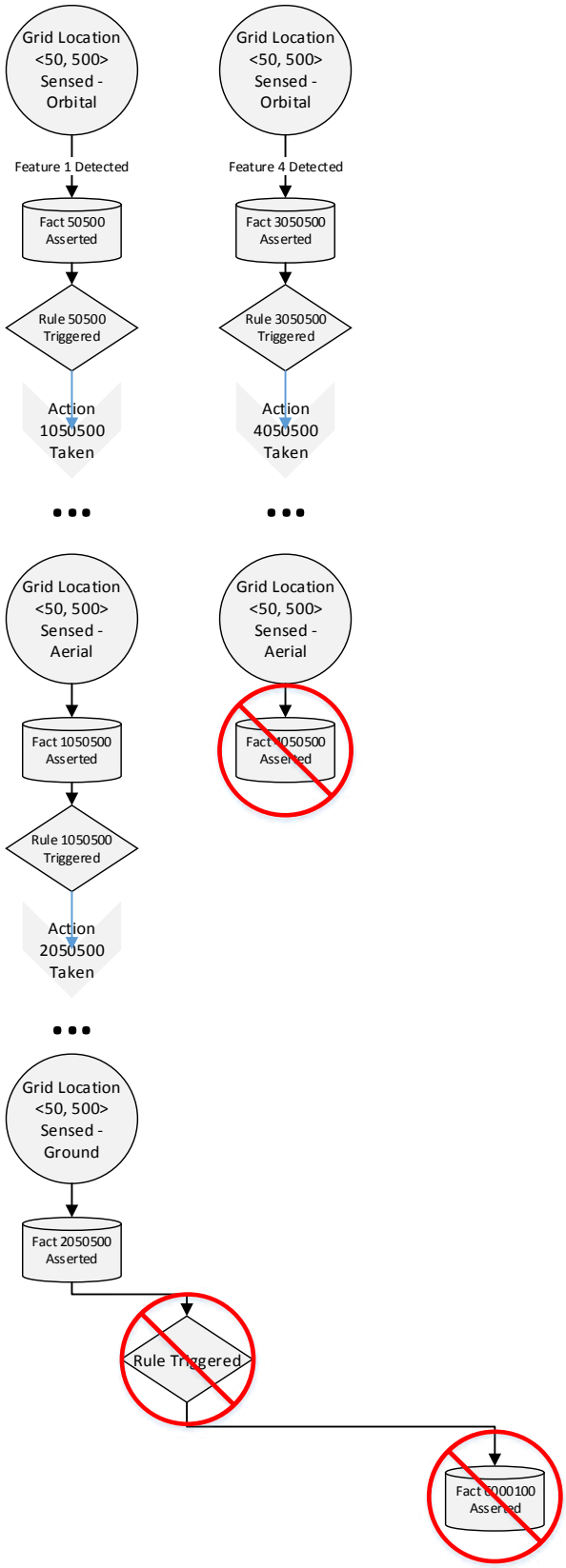

Figure 28. Failed operations of the Blackboard-based control network.

Robotic Command: Testing Under Typical and Atypical Conditions

Simulations were performed to assess the comparative efficacy of the two approaches (centralized and decentralized) under a variety of experimental conditions. To facilitate comparison, the same command methodology and software code was used (to the extent possible, excepting some code necessary for the specifics of each approach) for both the distributed and centralized methodologies. This same code base was used across all experimental conditions. Experimentation was conducted on a cluster of Intel i7 computers (each with 8 processor cores and 16 GB of RAM). Total command processing runtime (in milliseconds) and the amount of simulated time taken to complete each scenario are reported.

Simulations were conducted by creating a randomly-generated field of operations that is 1,000 x 1,000 grid locations in size (each grid location is nominally sized to correspond to a 100 meter x 100 meter area; however, for most purposes, the exact size of the grid locations is irrelevant, as the command decision making algorithm would work similarly across multiple grid sizes). Experimental conditions are created by controlling the frequency of several categories of scenario elements: map features (of multiple types) of interest and the rates of data collection errors, communications errors, temporary craft incapacitation and permanent craft incapacitation. For each run of the experiment, a new map is generated, a new scenario file (corresponding to the occurrences of various simulated error conditions) is generated, a corresponding Blackboard-style network is generated and the simulation is run.

Comparison of Approaches Under Error Conditions

The first area of data collection and analysis was the performance of the system

under normal and various error conditions. These error conditions simulated temporary and ongoing periods of system incapacitation. They included data collection error, communication error, temporary craft incapacitation and permanent craft incapacitation. Performance of the two systems under the various conditions was compared and a statistical t-test was applied to assess statistical significance. A one-tailed t-test was calculated for all conditions (based on the nominal thesis that the distributed system would outperform the centralized system). The processing times, scenario completion times and t-values are presented in Table 27 and Table 28.

As might be expected, no statistically significant (at $p < 0.10$) difference was experienced under the error-free condition, for either processing time or scenario completion time. Statistically significant (at $p < 0.10$) out performance of the distributed approach was demonstrated for the communications error and temporary incapacitation conditions, in terms of the number of turns required to complete the scenario. The distributed approach under-performed for the data collection error and permanent incapacitation scenarios, in terms of scenario completion (violating the premise of the one-tailed t-test that was conducted) and, thus, a two-tailed t-test was used to assess the statistical significance of the difference in performance. For the data collection error scenario, a significant difference in performance was identified, showing that the centralized approach may be more resilient to this type of error. This will serve as a prospective topic for future study. The difference in performance under the permanent incapacitation scenario was not shown to be significant at $p < 0.10$ for this data set; however the prospective efficacy of the centralized approach for this condition also merits further review.

Table 27. Processing Time and T-Value for Various Error Conditions (in ms).

Condition	Processing Time				T-Value
	Centralized		Distributed		
	Mean	Median	Mean	Median	
No Error	292388.95	228494	321729.7	295468	-
Data Collection Error (20)	287998.55	219042	270079.7	226006	0.34
Communication Error (20)	355701.5	284873.5	397089.3	284019	-
Temporary Incapacitation (10)	419561.4	299730	304835.9	261264	0.08
Permanent Incapacitation (10)	318422.65	265797.5	316975	240180.5	0.98

Table 28. Scenario Completion Time and T-Value for Various Error Conditions (in turn-units).

Condition	Scenario Completion Time				T-Value
	Centralized		Distributed		
	Mean	Median	Mean	Median	
No Error	88	85	87.5	72.5	0.48
Data Collection Error (20)	84.25	65	119.25	115	0.08*
Communication Error (20)	126	100	92.25	67.5	0.07
Temporary Incapacitation (10)	111.25	92.5	82.75	60	0.07
Permanent Incapacitation (10)	98.75	97.5	139.75	127.5	0.10*

* Two-tailed t-test

For four of the five scenarios, no statistically significant difference was detected between the centralized and distributed approaches in terms of processing time. A significant (at $p < 0.10$) difference was detected in terms of responding to temporary incapacitation.

Performance under several experimental conditions which combined the simulation of multiple types of error was then conducted. These are presented in Table 29 and Table 30. In all cases (excepting the no error case), the distributed approach out-performed the centralized one in terms of processing time. However, this difference was only significant (at $p < 0.10$) in the case of the combined temporary and permanent incapacitation scenario.

In terms of scenario completion time, the distributed approach, again, outperformed in all areas except for one (combined communications error and incapacitation). None of these differences in performance (including a two-tailed assessment of the difference in the

area where the distributed approach didn't outperform the centralized one) was statistically significant at $p < 0.10$.

Table 29. Processing Time and T-Value for Combined Error Conditions (in ms).

Condition	Processing Time				T-Value
	Centralized		Distributed		
	Mean	Median	Mean	Median	
No Error	292389	228494	321729.7	295468	-
Temporary (10) & Permanent (10) Incapacitation	423331.1	398438.5	268171.7	214913	0.009
Communications Error (20) & Incapacitation (10/10)	329056.6	235045.5	297672.7	217480	0.32
All Errors (Comm./Coll. = 20; Incap. = 10)	405592.8	360067.5	364212.6	318271.5	0.26

Table 30. Scenario Completion Time and T-Value for Combined Error Conditions (in turn-units).

Condition	Scenario Completion Time				T-Value
	Centralized		Distributed		
	Mean	Median	Mean	Median	
No Error	88	85	87.5	72.5	0.49
Temporary (10) & Permanent (10) Incapacitation	123.75	107.5	106.5	87.5	0.22
Communications Error (20) & Incapacitation (10/10)	87.75	72.5	91.25	70	0.86*
All Errors (Comm./Coll. = 20; Incap. = 10)	76.3	62.5	69.15	56.5	0.36

* Two-tailed t-test

Summary

This chapter has presented the research contribution of comparing the performance of centralized and decentralized command approaches under normal operating and impaired conditions. It has demonstrated the approximate equivalency (non-statistically significant difference) of the centralized and decentralized command approaches under normal operating conditions and how the impairment conditions affect the two command approaches differently.

Multiple statistically significant findings (at $p < 0.10$) were recorded through this process of assessment. The decentralized approach was shown to have (statistically significant) faster processing time for temporary craft incapacitation. The decentralize approach was also shown to have faster scenario completion time for communications error

and temporary craft incapacitation. The centralized approach was shown to have faster scenario completion time under the data collection error scenario. The decentralized command approach was also shown to have faster processing time under scenarios that combined both temporary and permanent craft incapacitation. No statistically significant findings were generated for scenario completion time for combined error conditions.

The foregoing demonstrates that the type of interference and/or other risk factors applicable to a given application play a significant role in the determination of what type (centralized or distributed) of command strategy to select for the mission. Neither approach is an across-the-board best decision; thus, the prospective likelihood, frequency and severity of events that could cause each of the various types of impairments should be taken into account when selecting a mission command strategy.

CHAPTER IX

CONCLUSIONS AND FUTURE WORK

The proceeding chapters have each discussed the design, development and characterization of critical elements of a system for commanding heterogeneous craft. Chapter 1 provided an overview of the work, the research question and the key questions that the work sought to answer. Chapter 2 presented prior related work. In Chapter 3, focus turned to the development of a distributed Blackboard Architecture-based system for robotic control. Chapter 4 presented an overview of the research methodology.

Chapters 5 to 8 presented work on the characterization of various components of the system. Chapter 5 discussed the use of pruning on blackboards and the benefits that it provides. Chapter 6 applied this pruning to a long-running robotic control system. In Chapter 7, focus turned to the implementation of the distribution of knowledge and the use of boundary nodes was proposed. Chapter 8 spoke to the key question of this work: characterizing circumstances under which centralized or distributed control would outperform each other.

Summary of Contributions

The work presented in these chapters has considered multiple approaches to conducting multi-craft missions for craft with heterogeneous capabilities (a number of which apply to, but may not be needed in, the simpler case of commanding a collection of

homogeneous robots). The work, thus, has made a number of contributions to the discipline. First, it has applied the pre-existing Blackboard Architecture to this command challenge. A variety of logistical and development challenges were solved in this process.

Second, the basic Blackboard Architecture concept has been expanded to support mission-driven operations through the addition of a solver mechanism. The solver changes the traditional forward-chaining approach to Blackboard operations (where conclusions are drawn from information provided and actions are potentially triggered by operating principles embodied in rules) to a data and goal driven methodology. Under this paradigm, rules give context to the data (instead of being created with a particular type of operation in mind) allowing the system to expand beyond its originally intended area of use. The solver attempts to find pathways to support or refute conclusions of interest through data collection and analysis operations. Multiple solver approaches and their comparative merits were assessed.

Third, several key additions were made to the Blackboard Architecture to support distributed and long-term robotic operations. Boundary nodes, extending existing work on boundary objects, are an integral part of making a system that is locally-responsive while being globally aware and able to communicate over limited bandwidth connections. In addition to their utility for robotic control, other subsequent work [0] has demonstrated their prospective efficacy for multi-homed online system control.

Forth, a key operating issue has been resolved. Operation in a real-world environment over any extended period of time presents a problem of data overload. The system is either forced to arbitrarily discard information (without knowing its importance) or become bogged down by the ever-growing data set. Pruning was applied to the

Blackboard Architecture and, subsequently, considered in the context of a robotic mission as a solution to this problem.

Finally, information was collected to help answer a key design question in robotic command: whether centralized or distributed control was most effective for normal operations and a variety of impairment scenarios. This expands the existing knowledge in this area which, previously, was based on a non-validated design assumption by Fink related to the selection of a centralized architectural approach.

Key Findings

A key goal of the work presented herein is to provide information to system designers to inform design decisions for heterogeneous multi-robot systems. Several results of this work are directly responsive to this goal.

The characterization of the pruning process demonstrated the efficacy and value of the use of pruning. Pruned networks were shown to require less than one-half of the time-to-solve of non-pruned networks. Moreover, the comparative cost of solving and pruning were considered. For networks similar to the one used for testing, approximately 150 solver uses (typically the solver is run repetitively as new information is added or information is updated on the Blackboard) would be required to justify the time-cost of pruning. Of course, the fact that pruning can be done at convenient times (when the system's processors are not otherwise needed) means that pruning may be adopted for its real-time / near real-time performance benefits alone. Re-pruning was also shown to be much less computationally intensive than initial pruning. The effect of pruning on system longevity was also demonstrated.

The utility of boundary node-based data encapsulation and replication has also been demonstrated. Using the boundary nodes, data transmission needs were reduced by two orders of magnitude from the use of a centralized blackboard approach and were about 60% less to one-third of the data transmission requirements of a hierarchical approach. The proposed approach also consistently outperformed limited and full replication strategies. Boundary nodes, thus, have been shown to be a key way to reduce communications needs. In addition to demonstrating their efficacy, this demonstrate communications reduction potential may be a key factor in command architecture selection decisions for many (communications constrained) missions.

Finally, the efficacy of the distributed and centralized command approaches was demonstrated. The distributed approach was shown to perform roughly equivalent to the centralized approach under many scenarios. However, in the case of communications errors and temporary craft incapacitation, it was shown to reduce scenario completion time by a statistically ($p<0.10$) and practically significant amount. It was also shown to reduce processing time for temporary craft incapacitation and combined temporary/permanent craft incapacitation scenarios by a statistically and practically significant amount. The centralized command approach, conversely, was shown to provide practically and statistically significant superior performance for completion time under data collection error scenarios and approached statistical significance (with a practically significant difference in result) for completion time for permanent craft incapacitation scenarios.

Considering Pruning and Command Strategy Selection

Previous chapters have discussed the impact of utilizing pruning (Chapters V and

VI) and the comparative performance of centralized versus decentralized command strategies (Chapter VIII). The efficacy of using pruning in any given application is driven by a number of application-specific factors that determine what level of prunable rules, facts and actions are present and the impact of their presence on network operations. The work in Chapters V and VI assessed this impact in terms of randomly generated networks, to provide a general-purpose heuristic that could serve the process of initial decision making. However, a final decision is more nuanced.

The networks generated in Chapters V and VI initially had a significant number of immediately prunable nodes. This, however, is more typical of an exploration system's network at later points, once significant data has been collected, rendering parts of the network irrelevant (as they would only be activated by the assertion of facts that are now known to be false). Pruning the networks presented in Chapter VIII before system operations would not result in a significant level of removal (depending on the settings for the potentially network-operations-impactful 'unlikely to be needed' pruning, it may result in no removal at all).

At later points in network operations, pruning may be more helpful. However, given the typical prioritization of system operations processing over data processing, the impact would likely not be on mission completion time (unless the data processing being potentially displaced was required for mission completion) but on the potential to do scientific analysis onboard (potentially being most impactful to secondary and tertiary goals, and not to the primary one). The prospective benefits of pruning might also be considered in the context of processor sizing, where the 50% reduction in pruning might facilitate the use of a lower-cost, lower-mass and/or lower-volume processor, reducing

overall mission cost levels.

Because of the parallel construction of the blackboard networks, the impact of pruning would be similar between centralized and distributed command approaches. Boundary nodes would typically not be prunable, as they represent higher-level data abstractions. Pruning would have some impact on the comparative performance on different types of data replication. The use of pruning might remove nodes that would otherwise be replicated under full replication, limited replication and central blackboard configurations (these nodes could be changed despite the fact that the change is irrelevant to future network operations). The impact, here, would be highly dependent on blackboard network design. However, as full and limited replication are not viable for most scenarios (as they require the nodes to be fully connected in a way that supports multicast traffic) and the central blackboard approach is two-to-three orders of magnitude more transmission-expensive than the proposed and hierarchical solutions, the prospective impact of pruning won't be a major consideration in replication configuration, for most applications.

Future Work

Several areas for prospective future work are indicated by the work that has been presented herein. First, as was previously identified, further assessment of the performance of the system under different levels of error conditions may yield other indications of areas of prospective differences in performance. Second, conditions which may be specific to various operating scenarios (such as dramatically difference movement conditions in certain areas of the operating region) should be assessed to determine what impact these

may have on the comparative performance of the two command approaches. The identification of additional comparative differences may inform, more granularly, the selection of a given approach for real-world missions with some or all of these characteristics present.

Third, the technologies developed for this work may have application to other areas of research (and real-world use) beyond the application described herein. An exploration of these prospective additional uses may drive future work in several areas.

Fourth, the characterization of the impact of pruning on multiple forms of the blackboard decision-making rule-fact-action networks remains a topic for future work. Two key areas of work are prospectively interesting, in this area. The first is the characterization of the impact of pruning on changing networks. Specifically, the impact of pruning on a network that is concurrently changing while the pruner runs and that is solving as the pruner is running and as the network is changing between prunings would provide additional insight into the efficacy of the pruner's use for craft where the data collection capability to processing capability ratio is higher than was simulated herein.

The simulation of this would test several independent variables: multiple (3) speeds of pruning, multiple test durations (e.g., 1,000, 5,000, 10,000, 20,000 and 50,000 turns), the impact of beginning pruning at multiple points (100, 250, 500, 1,000, 5,000) during longer duration tests and multiple simulation area (and thus blackboard network) sizes (e.g., 1,000 x 1,000, 2,000 x 2,000, 5,000 x 5,000). For each, the duration to first result, the average number of results and the total computational time required would be collected and recorded. Tests of statistical significance would then be applied to each of the 225

experimental conditions to assess the comparative impact of pruning during the applicable mission.

The second are of prospective interest, relative to Blackboard network pruning, would be to conduct static network tests across the experimental conditions listed above. This would eliminate any potential confounding of the data caused by the concurrent occurrence of data changes and pruning. The juxtaposition of these two result sets (using the same experimental conditions), using statistical significance testing, would facilitate the determination of the impact of concurrent pruner-solver operations. Demonstrating that this works (or does not work) well would inform the mission design of future prospective missions.

Fifth, the testing of the impact of pruning on the two different command strategies and multiple replication strategies is another area of interest. Based on the results of the long-running simulation testing described above, several conditions (with specific variable combinations for pruning speed, multiple test durations, point of pruning and simulation area size) could be selected to serve as independent variables in conjunction with a choice of command architecture (centralized or distributed). In the context of the distributed command architecture selection, each of the five data transmission / synchronization strategies (boundary node, full replication, limited replication, central blackboard and hierarchical) could also be tested. Presuming that three long-running simulation configurations were selected to serve as an independent variable (along with the six command architecture / data transmission / synchronization strategy choices), this would generate 18 experimental conditions. This data could then be analyzed using statistical

significance testing to ascertain the impact of using these different options in system design.

Finally, the validation of the experimentation performed via simulation through a real-world test mission is required to advance the Technology Readiness Level (TRL) to a point where the control technology would be deemed suitable for future work. This large-scale endeavor may identify other characteristics that may differentiate the performance of the distributed and centralized control approaches.

APPENDICES

Appendix A
Glossary of Terms

Central Analysis, Planning and Tasking – A system module that is responsible for high-level system planning.

Globally Unique Identifier – A value generated in a manner such that the chance of duplication is extremely low.

Ground Rover – A robot that operates on the surface of the Earth or another planet.

Micro-Aerial Vehicles – Unmanned aerial vehicles of a small size (typically small enough to fit in a human hand.

Multi-Blackboard System – A system that utilizes multiple agents, each with their own Blackboard for decision making.

Multi-Tier Autonomous Mission Architecture – The presented approach for controlling a mission comprised of orbital, aerial and ground craft.

Null Hypothesis – an assertion of current status that can be rejected through assessment of statistical significance.

Remotely Piloted Vehicle – An unmanned aerial vehicle that is controlled by a human from a remote location.

Technology Readiness Level – A system for evaluating the current status of a technology or system to facilitate the assessment of it for missions being planned.

Ticks - Ticks are the smallest unit of time measured by the Windows operating system [101]. A tick is equal to 100 nanoseconds.

Appendix B
Glossary of Acronyms / Abbreviations

CAPaT – Central Analysis, Planning and Tasking

GUID – Globally Unique IDentifier

H0 – null hypothesis

MAVs – Micro-Aerial Vehicles

MBS - Multi-Blackboard System

MTAMA – Multi-Tier Autonomous Mission Architecture

RAM – Random Access Memory

RPV – Remotely Piloted Vehicle

TRL – Technology Readiness Level

UAV – Unmanned Aerial Vehicle

UAS – Unmanned Aerial System

REFERENCES

[1] W. Fink. Generic prioritization framework for target selection and instrument usage for reconnaissance mission autonomy. Presented at Neural Networks, 2006. IJCNN'06. International Joint Conference On. 2006, .

[2] W. Fink, J. M. Dohm, M. A. Tarbell, T. M. Hare, V. R. Baker, D. Schulze-Makuch, R. Furfaro, A. G. Fairén, T. Ferre and H. Miyamoto. Tier-scalable reconnaissance missions for the autonomous exploration of planetary bodies. Presented at Aerospace Conference, 2007 IEEE. 2007, .

[3] K. Durga Prasad and S. Murty. Wireless sensor Networks–A potential tool to probe for water on moon. *Advances in Space Research 48(3),* pp. 601-612. 2011.

[4] E. Vassev, M. Hinchey and J. Paquet. Towards an ASSL specification model for NASA swarm-based exploration missions. Presented at Proceedings of the 2008 ACM Symposium on Applied Computing. 2008, .

[5] J. Straub. Spatial computing in an orbital environment: An exploration of the unique constraints of this special case to other spatial computing environments. Presented at Proceedings of the 2013 Spatial Computing Workshop at the Autonomous Agents and Multi-Agent Systems (AAMAS) 2013 Conference. 2013, .

[6] P. Brownell. The motivational impact of management-by-exception in a budgetary context. *Journal of Accounting Research 21(2),* pp. 456-472. 1983.

[7] S. W. Dekker and D. D. Woods. To intervene or not to intervene: The dilemma of management by exception. *Cognition, Technology & Work 1(2),* pp. 86-96. 1999.

[8] J. Straub. Model based data transmission: Analysis of link budget requirement reduction. *Communications and Network 4(4),* pp. 278-287. 2012.

[9] P. Troutman, D. D. Mazanek, F. Stillwagen, J. Antol, T. R. Sarver-Verhey, D. J. Chato, R. J. Saucillo, D. R. Blue, D. Carey and S. A. Krizan. Orbital aggregation and space infrastructure systems (OASIS). Presented at 53rd International Astronautical Congress–World Space Congress, Houston, TX. 2002, .

[10] R. Castano, M. Judd, T. Estlin, R. C. Anderson, D. Gaines, A. Castaño, B. Bornstein, T. Stough and K. Wagstaff. Current results from a rover science data analysis system. Presented at Aerospace Conference, 2005 IEEE. 2005, .

[11] C. Tsatsoulis. Sensor webs as multiagent, negotiating systems. Presented at NASA Earth Science Technology Conference. 2008, .

[12] A. Talukder, A. V. Panangadan, N. Georgas, T. Herrington and A. F. Blumberg. Integrated operational control of unattended distributed coastal sensor web systems with mobile autonomous robots. *Selected Topics in Applied Earth Observations and Remote Sensing, IEEE Journal Of 3(4),* pp. 442-450. 2010.

[13] W. Z. Song, B. Shirazi, S. Kedar, S. Chien, F. Webb, D. Tran, A. Davis, D. Pieri, R. LaHusen and J. Pallister. Optimized autonomous space in-situ sensor-web for volcano monitoring. Presented at Aerospace Conference, 2008 IEEE. 2008, .

[14] S. Chien, D. Tran, J. Doubleday, A. Davies, S. Kedar, F. Webb, G. Rabideau, D. Mandl, S. Frye and W. Song. A multi-agent space, in-situ volcano sensorweb. Presented at International Symposium on Space Artificial Intelligence, Robotics, and Automation for Space (i-SAIRAS 2010). Sapporo, Japan. 2010, .

[15] C. Zhong and S. A. DeLoach. Runtime models for automatic reorganization of multi-robot systems. Presented at Proceedings of the 6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. 2011, .

[16] N. E. Leonard and E. Fiorelli. Virtual leaders, artificial potentials and coordinated control of groups. Presented at Decision and Control, 2001. Proceedings of the 40th IEEE Conference On. 2001, .

[17] P. Dasgupta. "Multi-robot task allocation for performing cooperative foraging tasks in an initially unknown environment," in *Innovations in Defence Support Systems-2*Anonymous 2011, .

[18] M. Al-Khawaldah, O. Badran and I. Al-Adwan. Exploration algorithm technique for multi-robot collaboration. *Jordan Journal of Mechanical and Industrial Engineering* pp. 177-184. 2011.

[19] H. Chen, X. Wang and Y. Li. A survey of autonomous control for UAV. Presented at Artificial Intelligence and Computational Intelligence, 2009. AICI'09. International Conference On. 2009, .

[20] H. Oh, D. Won, S. Huh, D. H. Shim, M. Tahk and A. Tsourdos. Indoor UAV control using multi-camera visual feedback. *Journal of Intelligent & Robotic Systems 61(1-4),* pp. 57-84. 2011.

[21] T. Hester and P. Stone. Negative information and line observations for monte carlo localization. Presented at Robotics and Automation, 2008. ICRA 2008. IEEE International Conference On. 2008, .

[22] T. Bailey, M. Bryson, H. Mu, J. Vial, L. McCalman and H. Durrant-Whyte. Decentralised cooperative localisation for heterogeneous teams of mobile robots. Presented at Robotics and Automation (ICRA), 2011 IEEE International Conference On. 2011, .

[23] W. Zhu and S. Choi. An auction-based approach with closed-loop bid adjustment to dynamic task allocation in robot teams. Presented at Proceedings of the World Congress on Engineering. 2011, .

[24] E. M. Hinman and D. M. Bushman. Soviet automated rendezvous and docking system overview. Presented at Automated Rendezvous and Capture Review. Executive Summary. 1991, .

[25] K. Young, "US Achieves Autonomous Docking In Space," *New Scientist,* May 8 2007, 2007.

[26] S. Chien, R. Doyle, A. G. Davies, A. Jonsson and R. Lorenz. The future of ai in space. *Intelligent Systems, IEEE 21(4),* pp. 64-69. 2006.

[27] (November 1 2009). *Autonomous Sciencecraft Experiment*. Available: http://ase.jpl.nasa.gov/.

[28] (November 23 2009). *How does Remote Agent work?*. Available: http://www.qrg.northwestern.edu/projects/vss/docs/Remote-agent/zoom-how.html.

[29] (October 6 2004). *NASA Software Enables Satellites to Troubleshoot in Space*. Available: http://www.nasa.gov/vision/earth/lookingatearth/software_eo1_prt.htm.

[30] D. Bernard, R. Doyle, E. Riedel, N. Rouquette, J. Wyatt, M. Lowry and P. Nayak. Autonomy and software technology on NASA's deep space one. *Intelligent Systems and their Applications, IEEE 14(3),* pp. 10-15. 1999.

[31] (November 23 2009). *Hayabusa* [http://nssdc.gsfc.nasa.gov/nmc/spacecraftDisplay.do?id=2003-019A].

[32] P. Davies and J. Barrington-Cook. The impact of autonomy on the onboard software for the rosetta mission. Presented at Data Systems in Aerospace-DASIA 97. 1997, .

[33] (2005). *Deep Impact: The First Look inside a Comet*. Available: http://www2.ifa.hawaii.edu/newsletters/article.cfm?a=200&n=1.

[34] D. G. Kubitschek. Impactor spacecraft encounter sequence design for the deep impact mission. 2005.

[35] (November 23 2009). *What are the advantages of remote agents*. Available: http://www.qrg.northwestern.edu/projects/vss/docs/Remote-agent/1-advantages.html.

[36] (November 7 2009). *Autonomous Remote Agent* [http://nmp.nasa.gov/ds1/tech/autora.html].

[37] J. Schlecht, K. Altenburg, B. M. Ahmed and K. E. Nygard. Decentralized search by unmanned air vehicles using local communication. Presented at Proceedings of the International Conference on Artificial Intelligence. 2003, .

[38] C. A. Lua, K. Altenburg and K. E. Nygard. Synchronized multi-point attack by autonomous reactive vehicles with simple local communication. Presented at Swarm Intelligence Symposium, 2003. SIS'03. Proceedings of the 2003 IEEE. 2003, .

[39] D. Schesvold, J. Tang, B. M. Ahmed, K. Altenburg and K. E. Nygard. POMDP planning for high level UAV decisions: Search vs. strike. Presented at In Proceedings of the 16th International Conference on Computer Applications in Industry and Engineering. 2003, .

[40] N. Michael, E. Stump and K. Mohta. Persistent surveillance with a team of mavs. Presented at Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference On. 2011, .

[41] G. Punzo, G. Dobie, D. J. Bennet, J. Jamieson and M. Macdonald. Low-cost, multi-agent systems for planetary surface exploration. Presented at Proceedings of the 63rd International Astronautical Congress. 2012, .

[42] M. Herbert, E. Krotkov and T. Kanade, "A perception system for a planetary explorer," in *Proceedings of the 28th Conference on Decision and Control,* Tampa, FL, 1989, pp. 1151-1156.

[43] J. Bares, M. Hebert, T. Kanade, E. Krotkov, T. Mitchell, R. Simmons and W. Whittaker. Ambler: An autonomous rover for planetary exploration. *Computer 22(6),* pp. 18-26. 1989.

[44] R. T. Newton and Y. Xu. Neural network control of a space manipulator. *Control Systems, IEEE 13(6),* pp. 14-22. 1993.

[45] C. Thorpe, D. Wettergreen and R. Whittaker, "Dante's expedition to mount erebus," in *Robotics Research Reviews*Anonymous School of Computer Science, Carnegie Mellon University, 1992, .

[46] D. Wettergreen and C. Thorpe. Developing planning and reactive control for a hexapod robot. Presented at Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference On. 1996, .

[47] (2009). *Dante II*. Available: http://www.ri.cmu.edu/research_project_detail.html?project_id=163&menu_id=261.

[48] S. Hayati, R. Volpe, P. Backes, J. Balaram, R. Welch, R. Ivlev, G. Tharp, S. Peters, T. Ohm and R. Petras. The rocky 7 rover: A mars sciencecraft prototype. Presented at Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference On. 1997, .

[49] (2009). *Lunar Rover Navigation 1997*. Available: http://www.cs.cmu.edu/~lri/nav97.html.

[50] S. Moorehead, G. Simmons, D. Apostolopolous and W. Whittaker. Autonomous navigation field results of a planetary analog robot in antarctica. Presented at Artificial Intelligence, Robotics and Automation in Space. 1999, .

[51] P. Tompkins, A. Stentz and W. Whittaker. Automated surface mission planning considering terrain, shadows, resources and time. Presented at Proceedings of i-SAIRAS 2001. 2001, .

[52] D. Wettergreen, B. Dias, B. Shamah, J. Teza, P. Tompkins, C. Urmson, M. Wagner and W. Whittaker. First experiment in sun-synchronous exploration. Presented at Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference On. 2002, .

[53] Carnegie Mellon Robotics Institute, "Field report april 27, 2003," Carnegie Mellon Robotics Institute, Pittsburgh, PA, April 27 2003. 2003.

[54] Carnegie Mellon Robotics Institute, "Field report august 31, 2004," Carnegie Mellon Robotics Institute, Pittsburgh, PA, August 31 2004. 2004.

[55] D. Wettergreen, M. Wagner, D. Jonak, V. Baskaran, M. Deans, S. Heys, D. Pane, T. Smith, J. Teza and D. Thompson. Long-distance autonomous survey and mapping in the robotic investigation of life in the atacama desert. Presented at International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS). 2008, .

[56] D. Wettergreen, D. Jonak, D. Kohanbash, S. Moreland, S. Spiker, J. Teza and W. Whittaker. Design and experimentation of a rover concept for lunar crater resource survey. Presented at 47th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition. 2009, .

[57] (December 26 2009). *Space Topics: Mars Missions to Mars* [http://www.planetary.org/explore/topics/mars/missions.html].

[58] H. Stone, "Mars pathfinder microrover A small, low-cost, low-power spacecraft," in *Proceedings of the 1996 AIAA Forum on Advanced Developments in Space Robotics,* 1996, .

[59] (January 23 2009). *In-situ Exploration and Sample Return: Autonomous Planetary Mobility*. Available: http://marsrovers.jpl.nasa.gov/technology/is_autonomous_mobility.html.

[60] (February 13 2007). *Feb. 13: Carnegie Mellon Software Steers NASA's Mars Rover* [http://www.cmu.edu/news/archive/2007/February/feb13_rovers.shtml].

[61] J. L. Bresina and P. H. Morris. Mixed-initiative planning in space mission operations. *AI Magazine 28(2),* pp. 75. 2007.

[62] (2009). *ASPEN*. Available: http://aspen.jpl.nasa.gov.

[63] G. Rabideau, R. Knight, S. Chien, A. Fukunaga and A. Govindjee. Iterative repair planning for spacecraft operations using the aspen system. Presented at Artificial Intelligence, Robotics and Automation in Space. 1999, .

[64] T. Smith, R. Simmons, S. Singh and D. Hershberger. Future directions in multi-robot autonomy and planetary surface construction. Presented at Proceedings of the 2001 Space Studies Institute Conference. 2001, .

[65] (2009). *CASPER*. Available: http://casper.jpl.nasa.gov.

[66] T. Estlin, G. Rabideau, D. Mutz and S. Chien. Using continuous planning techniques to coordinate multiple rovers. *Electronic Transactions on Artificial Intelligence 4(45-57),* pp. 2000. 2000.

[67] S. Chien, B. Engelhardt, R. Knight, G. Rabideau and R. Sherwood. Onboard autonomy on the three corner sat mission. Presented at 6th International Symposium on Artifical Intelligence, Robotics and Automation in Space. 2001, .

[68] F. Fisher, T. Estlin, D. Gaines, S. Schaffer, C. Chouinard and R. Knight. CLEaR: Closed loop execution and recovery-a framework for unified planning and execution. 2002. Available: http://hdl.handle.net/2014/10168.

[69] (2009). *Onboard Autonomous Science Investigation System*. Available: http://oasis.jpl.nasa.gov/index.html.

[70] B. D. Smith, B. E. Engelhardt, D. H. Mutz and J. P. Crawford. Automated planning for the modified antarctic mapping mission. Presented at Aerospace Conference, 2001, IEEE Proceedings. 2001, .

[71] K. D. Mullins, E. B. Pacis, S. B. Stancliff, A. B. Burmeister, T. A. Denewiler, M. H. Bruch and H. R. Everett, "An automated UAV mission system," United States Navy, 2003.

[72] K. L. Wagstaff, D. Mazzoni and S. Sain. HARVIST: A system for agricultural and weather studies using advanced statistical methods. 2005.

[73] R. Sherwood and S. Chien. Sensor web technologies: A new paradigm for operations. Presented at International Symposium on Reducing the Cost of Spacecraft Ground Systems and Operations (RCSGSO 2007). 2007, .

[74] R. Sherwood, S. Chien, D. Tran, B. Cichy, R. Castano, A. Davies and G. Rabideau. Autonomous science agents and sensor webs: EO-1 and beyond. Presented at Aerospace Conference, 2006 IEEE. 2006, .

[75] S. Chien, D. Tran, A. Davies, M. Johnston, J. Doubleday, R. Castano, L. Scharenbroich, G. Rabideau, B. Cichy and S. Kedar. Lights out autonomous operation of an earth observing sensorweb. Presented at Int. Symposium on Reducing the Cost of Spacecraft Ground Systems and Operations (RCSGSO 2007). 2007, .

[76] B. Hayes-Roth. A blackboard architecture for control. *Artif. Intell. 26(3),* pp. 251-321. 1985.

[77] L. D. Erman, F. Hayes-Roth, V. R. Lesser and D. R. Reddy. The hearsay-II speech-understanding system: Integrating knowledge to resolve uncertainty. *ACM Computing Surveys (CSUR) 12(2),* pp. 213-253. 1980.

[78] E. A. Feigenbaum, B. G. Buchanan and J. Lederberg. On generality and problem solving: A case study using the DENDRAL program. 1970.

[79] E. H. Shortliffe, R. Davis, S. G. Axline, B. G. Buchanan, C. C. Green and S. N. Cohen. Computer-based consultations in clinical therapeutics: Explanation and rule acquisition capabilities of the MYCIN system. *Computers and Biomedical Research 8(4),* pp. 303-320. 1975.

[80] M. V. Johnson Jr and B. Hayes-Roth. Integrating diverse reasoning methods in the BBP blackboard control Architecture1. Presented at Proceedings of the AAAI. 1987, .

[81] M. P. Georgeff and F. F. Ingrand. Monitoring and control of spacecraft systems using procedural reasoning. Presented at Proceedings of the Space Operations Automation and Robotics Workshop. 1989, .

[82] J. P. Rice. Poligon: A systems for parallel problem solving. Knowledge Systems Laboratory, Stanford University. Stanford, CA. 1986.

[83] D. D. Corkill, K. Q. Gallagher and P. M. Johnson. Achieving flexibility, efficiency, and generality in blackboard architectures. Presented at Proceedings of the National Conference on Artificial Intelligence. 1987, .

[84] J. Le Mentec and S. Brunessaux. Improving reactivity in a blackboard architecture with parallelism and interruptions. Presented at Proceedings of the 10th European Conference on Artificial Intelligence. 1992, .

[85] M. Hewett and R. Hewett. A language and architecture for efficient blackboard systems. Presented at Artificial Intelligence for Applications, 1993. Proceedings., Ninth Conference On. 1993, .

[86] G. Brzykcy, J. Martinek, A. Meissner and P. Skrzypczynski. Multi-agent blackboard architecture for a mobile robot. Presented at Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference On. 2001, .

[87] Y. Yang, Y. Tian and H. Mei. Cooperative Q learning based on blackboard architecture. Presented at International Conference on Computational Intelligence and Security Workshops, 2007. 2007, .

[88] R. E. Fayek, R. Liscano and G. M. Karam. A system architecture for a mobile robot based on activities and a blackboard control unit. Presented at Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference On. 1993, .

[89] A. M. de Campos and M. Monteiro de Macedo. A blackboard architecture for perception planning in autonomous vehicles. Presented at Industrial Electronics, Control, Instrumentation, and Automation, 1992. Power Electronics and Motion Control., Proceedings of the 1992 International Conference On. 1992, .

[90] S. Carroll, J. E. Boyd and J. Denzinger. Data-centered control of cooperating UAVs: Flying airplanes with a multimedia database. 2008.

[91] E. Shahbazian, J. Duquet and P. Valin. A blackboard architecture for incremental implementation of data fusion applications. Presented at Fusion. 1998, .

[92] D. Goldin and A. M. Chesnokov. Features of informational control complex of autonomous spacecraft. Presented at Proceedings of IFAC Workshop on Aerospace Guidance, Navigation and Flight Control Systems. 2011, .

[93] S. Chien, B. Cichy, A. Davies, D. Tran, G. Rabideau, R. Castano, R. Sherwood, D. Mandl, S. Frye and S. Shulman. An autonomous earth-observing sensorweb. *Intelligent Systems, IEEE 20(3),* pp. 16-24. 2005.

[94] R. H. Crites and A. G. Barto. Elevator group control using multiple reinforcement learning agents. *Mach. Learning 33(2-3),* pp. 235-262. 1998.

[95] W. Fink, J. M. Dohm, M. A. Tarbell, T. M. Hare and V. R. Baker. Next-generation robotic planetary reconnaissance missions: A paradigm shift. *Planet. Space Sci. 53(14),* pp. 1419-1426. 2005.

[96] J. Han, M. Li and L. Guo. Soft control on collective behavior of a group of autonomous agents by a shill agent. *Journal of Systems Science and Complexity 19(1),* pp. 54-62. 2006.

[97] K. Windt and M. Hülsmann. "Changing paradigms in logistics—understanding the shift from conventional control to autonomous cooperation and control," in *Understanding Autonomous Cooperation and Control in Logistics*Anonymous 2007, .

[98] (). *Mars, Water and Life*. Available: http://mars.jpl.nasa.gov/msp98/why.html.

[99] P. J. Denning, D. E. Comer, D. Gries, M. C. Mulder, A. Tucker, A. J. Turner and P. R. Young. Computing as a discipline. *Commun ACM 32(1),* pp. 9-23. 1989.

[100] J. Straub, "Comparing the Effect of Pruning on a Best-Path and Naive-Approach Blackboard Solver," *International Journal of Automation and Computing,* In Press.

[101] (). *TimeSpan.Ticks Property*.

[102] S. L. Star. This is not a boundary object: Reflections on the origin of a concept. *Science, Technology & Human Values 35(5),* pp. 601-617. 2010.

[103] M. Weiss and F. Stetter. A hierarchical blackboard architecture for distributed AI systems. Presented at Software Engineering and Knowledge Engineering, 1992. Proceedings., Fourth International Conference On. 1992, .

[104] J. Straub and H. Reza. The use of the blackboard architecture for a decision making system for the control of craft with various actuator and movement capabilities. Presented at Proceedings of the International Conference on Information Technology: New Generations. 2014, .