June 2022

Explainable and Cooperative Autonomy Across Networks of Distributed Systems

Peter Joseph Jorgensen
University of South Florida

Explainable and Cooperative Autonomy Across Networks of Distributed Systems

by

Peter Joseph Jorgensen

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Electrical Engineering
College of Engineering
University of South Florida

Date of Approval:
June 10, 2022

Keywords: Artificial Intelligence, Fuzzy Logic, Human-Centered Design, Satellite Constellation Management

Dedication

To Denise and Elliot, my foundation and my inspiration.

Acknowledgments

Writing this dissertation concludes what has been a long journey filled with challenges and uncertainty. I have been lucky to count so many caring and supportive individuals as trusted allies along the way.

My friends and family have been supportive of each challenge I undertook in pursuit of this achievement. Thank you for listening, asking questions, driving and flying across the country (many times), and encouraging me along the way.

To my friends throughout the University of South Florida, thank you for helping me understand USF and thrive as a member of the community. Your unwavering and selfless support made life as a graduate student easy.

To my teammates and coworkers at the Institute of Applied Engineering, your encouragement, flexibility, and top cover enabled me to finish this process. Thank you for always having my back and inspiring me to do my best.

To my committee members, thank you for providing guidance early on and helping scope my problem. Your complementary knowledge and experience made a robust ensemble of experts from which I drew inspiration. I am honored to have completed this dissertation under your supervision.

Thank you to my advisor, mentor, and major professor Robert Bishop. You trusted me to do the work. Navigating this process and working with you all these years has been an inspiration, an honor, and a pleasure.

Denise and Elliot, your patience, belief, encouragement, and understanding have been uncountably infinite. I could not have done this without you.

Table of Contents

List of Tables v

List of Figures vii

Abstract x

Chapter 1: Introduction 1
- 1.1 Contributions 2
- 1.2 Hierarchical Ensembles of Autonomous Decision Systems 3
- 1.3 Historical Foundations and Perspectives 4
- 1.4 Difficulty in Praxis, or Limitations of an Engineering Dissertation 5
- 1.5 Dissertation Structure and Motivation 6
  - 1.5.1 Chapter List 6
    - 1.5.1.1 Chapters 2-4: Foundations 6
    - 1.5.1.2 Chapters 5-7: HEADS Principles and Design 6
    - 1.5.1.3 Chapters 8-11: Practical Applications 7
  - 1.5.2 Alternative Reading Paths 7
  - 1.5.3 Motivation 8

Chapter 2: Foundations of Autonomy 9
- 2.1 Automation and Autonomy 9
- 2.2 Autonomous Agents 10
  - 2.2.1 Agent Theory 11
  - 2.2.2 Agent Architecture 11
    - 2.2.2.1 Logic Based Architecture 12
    - 2.2.2.2 Reactive Architecture 12
    - 2.2.2.3 Belief-Desire-Intent Architecture 13
    - 2.2.2.4 Layered Architecture 13
  - 2.2.3 Agent Language 13
- 2.3 Current Trends in Technologies Supporting Autonomy 14
  - 2.3.1 Technological Advancement 14
    - 2.3.1.1 Advanced Computing 14
    - 2.3.1.2 Advanced Components 15
    - 2.3.1.3 Advanced Communications 16
    - 2.3.1.4 Network Connectivity 17
  - 2.3.2 The Need for Verification and Validation 18

2.3.3 Artificial Intelligence and Machine Learning . . . 19
2.3.3.1 AI/ML and Autonomy . . . 20
2.4 Agent: A Working Definition . . . 21

Chapter 3: The Complex Nature of Autonomy . . . 23
3.1 Modeling Complexity . . . 24
3.2 Complex Agents as Complex System of Systems . . . 25
3.3 Networked and Distributed Systems-of-Systems (NDSS) . . . 25
3.3.1 A Representative NDSS . . . 27
3.4 Autonomy and Truth . . . 28

Chapter 4: Managing Complexity . . . 30
4.1 Probability and Uncertainty . . . 30
4.2 Fuzzy Theory . . . 32
4.2.1 Fuzzy Set Definition . . . 33
4.2.2 Rule Definition and Evaluation . . . 33
4.2.3 Aggregation and Defuzzification . . . 34
4.2.4 A Fuzzy Example . . . 36
4.3 Ensembles . . . 37
4.4 Hierarchies . . . 39
4.5 Measuring Performance . . . 40
4.6 Notes on Computational Complexity . . . 41
4.7 Notes on Human Factors . . . 44
4.7.1 Language Is Human . . . 45
4.7.2 Linguistic Variables, Complexity, and Understandability . . . 46

Chapter 5: Hierarchical Ensembles of Autonomous Decision Systems . . . 49
5.1 Principles and Attributes . . . 49
5.1.1 Human-Centered Design . . . 49
5.1.2 Bounded Behavior . . . 50
5.1.3 System Modularity . . . 51
5.2 Structure . . . 51
5.2.1 Signals . . . 51
5.2.2 Ensembles . . . 53
5.2.3 Expert Systems . . . 54
5.2.4 Hierarchies . . . 57
5.3 HEADS as an Adaptable Framework . . . 57
5.3.1 Expert System Selection . . . 57
5.3.2 Heterogeneous Distribution of Experts . . . 58
5.3.3 Parameterization . . . 59
5.3.4 Redundancy and Overlap . . . 59
5.4 Notes on Agent Theory and Architecture . . . 60

Chapter 6: HEADS System Implementation and Analysis . . . 63
6.1 Define Early, Update Often . . . 63

6.2 Outward-In Approach 64
6.2.1 Defining the Signal Space 65
6.2.1.1 Inputs 65
6.2.1.2 Intermediate Outputs 65
6.2.1.3 Terminal Outputs 65
6.2.1.4 Signal Space Modularity 65
6.2.1.5 Recommendations for Identifying Signals 66
6.2.2 Defining Goal Based Operation 67
6.2.3 Defining Rule Activation Boundaries 68
6.3 Introspection: Signal Space Coverage 70
6.4 Introspection: System Complexity Metrics 72
6.5 Introspection: Sensitivity Analysis 75

Chapter 7: HEADS Performance and Optimization 77
7.1 Computational Performance 77
7.1.1 Resource Requirements 77
7.1.2 Computational Requirements 78
7.2 Decision Making Performance 79
7.3 Optimization 80
7.3.1 Optimizing for Human Factors 80
7.3.2 Optimizing for Performance 81
7.3.3 Evolutionary Optimization 81
7.3.3.1 HEADS as a Genetic Algorithm Problem 82
7.3.3.2 Genetic HEADS Primer 82

Chapter 8: HEADS Applied to the NDSS Problem 85
8.1 Simulation Design Overview 86
8.1.1 Orbital Remote Sensors 88
8.1.2 Network Communications 89
8.1.3 Goal Based Tasking 89
8.1.4 Simulated Failures 89
8.1.5 The NDSS: Spacecraft and Orbits 90
8.2 Simulation Performance Metrics 91
8.2.1 Spacecraft Performance 92
8.3 HEADS System Design 94
8.3.1 Signal Identification 95
8.3.2 Rule Base Definition 96
8.3.3 Fuzzy Set Parameterization 98
8.3.4 Design Introspection 99
8.3.4.1 Binary Coverage 99
8.3.4.2 Weighted Coverage 100
8.3.4.3 Complexity Metrics 101
8.3.4.4 Sensitivity Analysis 103

Chapter 9: Simulation Results and Performance Analysis 108

9.1 Sensor Constellation Performance 109
9.1.1 State: Energy and Power 109
9.1.2 State: Target Visible 110
9.1.3 State: Fault Detected 111
9.2 Performance Detail 112
9.2.1 Node B5 - Type 1 - Nominal 114
9.2.2 Node D34 - Type 2 - Nominal 117
9.2.3 Node A6 - Type 1 - Faulty 119
9.2.4 Node C28 - Type 2 - Faulty 121
9.3 Baseline Performance Conclusions 122

Chapter 10: Performance Optimization Approaches and Results 131
10.1 Manual Tuning Approach 131
10.2 Manual Tuning Results 133
10.3 Evolutionary Optimization Approach 134
10.3.1 Evolutionary Mechanisms 134
10.3.1.1 Constrained Evolution 136
10.3.1.2 Logical Ordering Constraints 137
10.4 Evolutionary Optimization Results 138
10.5 Conclusions 140

Chapter 11: Conclusions 142
11.1 Explainable Autonomy Is Valuable 142
11.2 HEADS Is Designer-Friendly 143
11.3 HEADS Offers Robust Performance 143
11.4 Performance Metrics Drive Performance 144
11.5 Future Work 144

References 146

Appendix A: Example HEADS Definitions and Parameterizations 157

Appendix B: Example HEADS Complexity Metrics 172

Appendix C: HEADS Performance Scores 179

About the Author . . . . End Page

List of Tables

Table 6.1 Example questions and answers for identifying the signal space 66

Table 8.1 Simulated spacecraft and orbit definitions 91

Table 8.2 Remote sensing resource use and scoring (type 1) 96

Table 8.3 Remote sensing resource use and scoring (type 2) 97

Table 8.4 Input and output signals for sensor node logic 97

Table 8.5 Rules for expert $s_{1,1}$ of ensemble y_1 98

Table 8.6 Fuzzy sets for expert $s_{1,1}$ of ensemble y_1 99

Table 8.7 Fuzzy set parameters for expert $s_{1,1}$ of ensemble y_1 99

Table 8.8 Example HEADS binary coverage 104

Table 8.9 Example HEADS weighted coverage 105

Table 8.10 Example HEADS complexity metrics for rules and fuzzy sets for ensemble y_1 105

Table 8.11 Example HEADS fuzzy set to signal mapping complexity for ensemble y_1 106

Table 8.12 Example HEADS rule similarity matrix for expert $s_{1,1}$ 106

Table 8.13 Example HEADS rule similarity matrix for expert $s_{1,2}$ 106

Table 9.1 Sensor nodes performance comparison 117

Table 10.1 Fuzzy set parameter updates from manual tuning process 132

Table A.1 HEADS example - baseline system rule definitions 157

Table A.2 HEADS example - baseline system fuzzy set parameters 160

Table A.3 HEADS example - manually-tuned rule updates 165

Table A.4 HEADS example - manually-tuned parameter updates 166

Table A.5 HEADS example - genetic tuning parameter updates 166

Table B.1 Example HEADS complexity metrics for rules and fuzzy sets 172

Table B.2 Example HEADS rule similarity matrix for expert $s_{1,1}$ 176

Table B.3 Example HEADS rule similarity matrix for expert $s_{1,2}$ 176

Table B.4 Example HEADS rule similarity matrix for expert $s_{2,1}$, $s_{3,1}$, and $s_{4,1}$ 176

Table B.5 Example HEADS rule similarity matrix for expert $s_{2,2}$, $s_{3,2}$, and $s_{4,2}$ 176

Table B.6 Example HEADS rule similarity matrix for expert $s_{2,3}$, $s_{3,3}$, and $s_{4,3}$ 176

Table B.7 Example HEADS rule similarity matrix for expert $s_{2,4}$, $s_{3,4}$, and $s_{4,4}$ 176

Table B.8 Example HEADS rule similarity matrix for expert $s_{5,1}$ 177

Table B.9 Example HEADS rule similarity matrix for expert $s_{5,2}$ 177

Table B.10 Example HEADS rule similarity matrix for expert $s_{6,1}$ 177

Table B.11 Example HEADS fuzzy set to signal mapping complexity 178

Table C.1 HEADS system performance - baseline and tuned comparison 179

Table C.2 HEADS system performance - baseline 181

Table C.3 HEADS system performance - manually-tuned 189

Table C.4 HEADS system performance - genetic tuning 196

List of Figures

Figure 4.1 A simple Markov model . . . 31

Figure 4.2 Fuzzy rule evaluation . . . 36

Figure 4.3 Fuzzy rule output aggregation and defuzzification . . . 37

Figure 4.4 Complexity reduction as a function of reduced input sub-space fraction for various input space sizes (constant output space size of $m = 1$) . . . 43

Figure 4.5 Complexity reduction as a function of reduced input sub-space fraction for various output space sizes (constant input space size of $n = 30$) . . . 44

Figure 5.1 Hierarchical Ensembles of Autonomous Decision Systems (HEADS) topology . . . 52

Figure 6.1 Fuzzy set activation functions . . . 69

Figure 8.1 Distribution of all spacecraft in Earth-centered inertial coordinates . . . 92

Figure 8.2 Distribution of communications spacecraft in Earth-centered inertial coordinates . . . 93

Figure 8.3 Distribution of processing spacecraft in Earth-centered inertial coordinates . . . 94

Figure 8.4 Distribution of remote sensing spacecraft in Earth-centered inertial coordinates . . . 95

Figure 8.5 Fuzzy set parameters for ensemble y_1, expert $s_{1,1}$, input signal stored_energy . . . 100

Figure 8.6 Fuzzy set parameters for ensemble y_1, expert $s_{1,1}$, input signal sunlit . . . 101

Figure 8.7 Fuzzy set parameters for ensemble y_1, expert $s_{1,1}$, input signal stored_energy . . . 102

Figure 8.8 Fuzzy set parameters for ensemble y_1, expert $s_{1,1}$, output signal self_good 103

Figure 8.9 Example HEADS sensitivity for ensemble y_1 expert $s_{1,2}$, output 'self_good', variable signal x_1 ('stored_energy') 106

Figure 8.10 Example HEADS sensitivity for ensemble y_1 expert $s_{1,2}$, output 'self_good', variable signal x_{11} ('fault_detected') 107

Figure 8.11 Example HEADS sensitivity for ensemble y_1 expert $s_{1,2}$, output 'self_good', variable signal y_6 ('fault_recovery') 107

Figure 9.1 Sensor node performance: stored energy state 110

Figure 9.2 Sensor node performance: stored energy state (detail of first four hours) 111

Figure 9.3 Sensor node performance: power differential 112

Figure 9.4 Sensor node performance: power differential (detail of first four hours) 113

Figure 9.5 Sensor node performance: target visible state 114

Figure 9.6 Sensor node performance: target visible state (detail of first four hours) 115

Figure 9.7 Faulty node (A6) and non-faulty node (B5) fault-detected state comparison 116

Figure 9.8 Node B5: payload activation and scoring 118

Figure 9.9 Node B5: ground target visibility 119

Figure 9.10 Node B5: processor nodes in range 120

Figure 9.11 Node B5: expert s_5_1 rule activation 121

Figure 9.12 Node B5: expert s_5_2 rule activation 122

Figure 9.13 Node B5: stored energy state 123

Figure 9.14 Node B5: total score over time (including battery, self good, and payload scoring) 123

Figure 9.15 Node B5: subtotal score over time (including payload scoring) 124

Figure 9.16 Node D34: payload activation and scoring 124

Figure 9.17 Node D34: ground target visibility 125

Figure 9.18 Node D34: communication and processor nodes in range 125

Figure 9.19 Node D34: stored energy state 126

Figure 9.20 Node D34: total score over time (including battery, self good, and payload scoring) 126

Figure 9.21 Node D34: subtotal score over time (including payload scoring) 127

Figure 9.22 Node A6: payload activation and scoring 127

Figure 9.23 Node A6: ground target visibility 128

Figure 9.24 Node A6: stored energy 128

Figure 9.25 Node A6: stored energy score and total score 129

Figure 9.26 Node A6: self_good sensitivity to stored energy status 129

Figure 9.27 Node C28: payload activation and scoring 130

Figure 9.28 Node C28: total score over time (including battery, self good, and payload scoring) 130

Figure 10.1 Stored energy state comparison between the baseline and tuned systems 134

Figure 10.2 Node D34 stored energy scoring comparison between the baseline and tuned systems 135

Figure 10.3 Node D34 sensor activation scoring comparison between the baseline and tuned systems 136

Figure 10.4 Node D34: payload activation and scoring for the tuned system 137

Figure 10.5 Unconstrained fuzzy set ordering for ensemble y_4 expert $s_{4,1}$ 139

Figure 10.6 Ensemble y_4 expert $s_{4,1}$ sensitivity to input signal 'stored_energy' 140

Figure 10.7 Activation of fuzzy set 'batt_good' has no impact on rule activation for ensemble y_4 expert $s_{4,1}$ 141

Figure 10.8 Stored energy comparison: manual and genetic tuned systems 141

Abstract

Large networks of complex systems-of-systems are commonplace and evermore present in both mundane and extraordinary facets of human existence. From the exponential growth of connectivity via the internet and other information networks, to the miniaturization of computers and sensors, to cross-domain sensor and communication networks, these networks of distributed systems-of-systems (NDSS) present incredible benefits and challenges. Autonomy is perhaps the most important and most difficult to achieve enabling technology for efficient performance of the NDSS. Giving each individual agent in a network the ability to manage its internal state in dynamic operating environments and in pursuit of multiple complex and possibly conflicting individual and network-level goals is key to efficient, robust, and adaptable performance. When humans are an integral part of the system through oversight, operation, physical interaction, or dependency, the safety, security, understandability, and verifiability of an autonomous agent's actions are paramount. These attributes and constraints describe a class of problems with members including autonomous ground vehicles operating on public motorways, autonomous air traffic control systems, constellations of Earth-observation satellites, and human-rated interplanetary spacecraft.

In this work, the problem of how to enable a high degree of on-board system autonomy for individual agents within the human-centered NDSS is explored. Foundations for agent-based autonomy are discussed in the context of understanding and managing complexity. A Hierarchical Ensembles of Autonomous Decision Systems (HEADS) framework is proposed. The HEADS framework is designed to enable cooperative, explainable, and robust multiple objective decision making by collections of fuzzy logic expert systems based on linguistic variables.

The HEADS framework is applied to a simulated constellation of Earth satellites tasked with managing sensor activation and maintaining battery status and subject to the dynamic space environment and fault conditions. The initial system design shows acceptable behavior and robustness against fault conditions. Performance and operational analysis provide the basis for manually tuning the system which increases performance by 48%. An additional 6.8% performance increase is seen after automated genetic algorithm based parameter tuning. The HEADS framework is shown to provide robust performance for the multiple objective dynamic decision making problem for a large network of distributed systems of systems.

Chapter 1: Introduction

Large networks of complex systems-of-systems are commonplace and evermore present in both mundane and extraordinary facets of human existence. From the exponential growth of connectivity via the internet and other information networks, to the miniaturization of computers and sensors [1], to cross-domain sensor and communication networks [2], these networks of distributed systems-of-systems (NDSS) present incredible benefits and challenges. Autonomy is perhaps the most important and most difficult to achieve enabling technology for efficient performance of the NDSS. Giving each individual agent in a network the ability to manage its internal state in dynamic operating environments and in pursuit of multiple complex and possibly conflicting individual and network-level goals is key to efficient, robust, and adaptable performance. When humans are an integral part of the system through oversight, operation, physical interaction, or dependency, the safety, security, understandability, and verifiability of an autonomous agent's actions are paramount [3, 4]. These attributes and constraints describe a class of problems with members including autonomous ground vehicles operating on public motorways, autonomous air traffic control systems, constellations of Earth-observation satellites, and human-rated interplanetary spacecraft.

In this work, the problem of how to enable a high degree of on-board system autonomy for individual agents within the human-centered NDSS is explored. Foundations for agent-based autonomy, including the challenges from and management of complexity, are discussed. A Hierarchical Ensembles of Autonomous Decision Systems (HEADS) framework designed to handle multiple objective decision making is proposed conceptually and through application to a representative NDSS problem.

This chapter discusses the contributions of this dissertation followed by a brief introduction to computational autonomy to provide context for the proposed framework. The human-centric nature of the problem is discussed. Limitations of making assertions about human factors are considered along with the literature-backed mitigation strategy employed in this work. There is a per-chapter map for the rest of the dissertation and alternate sub-maps for those who may be interested in either the theory or the application. Finally, the motivations for this work are presented.

1.1 Contributions

The primary contributions of this dissertation are:

1. the HEADS framework,
2. the HEADS design, analysis, and optimization process, and
3. HEADS applied to a representative NDSS problem.

The structure of the HEADS framework leverages multiple hierarchical collections of fuzzy experts to balance system performance and complexity. These cooperative-competitive fuzzy experts are the core logic agents and exhibit desirable characteristics including linguistic variable based reasoning and bounded outputs resulting in understandable and verifiable performance. This structure enables robust multiple-objective decision making.

The design methodology behind implementing HEADS includes a baseline approach, iteration, and introspective analysis. Top-level data modeling informs bottom-up design which supports subject matter experts in designing for problem domain specificity. The result is modular in that each component agent derives its expertise from domain specific knowledge. Framework analyses including signal space coverage, complexity metrics, and output sensitivity enable the system designer to manage understandability and performance. An approach to evolutionary optimization is presented for tuning the system according to cost/reward metrics.

The autonomous operation of a space-based Earth sensing network, a representative human-centered NDSS problem, is considered and a HEADS framework solution is designed and analyzed. Simulation results are shown before and after the system is optimized demonstrating generalized decision making. Simulation of partial and full agent failures and the resulting network-level performance further confirm robust performance.

Approaches to managing complexity and the adjacent concept of human factors are discussed in Chapter 4. As intrinsically related to the motivations of this dissertation, the human-centered design approach has informed many of the specific design decisions behind the HEADS framework. Secondary contributions of this dissertation include an overview of the foundations of autonomy through the lens of agents and recent trends in artificial intelligence in Chapter 2 and a discussion on the nature of complexity as applied to agent-based autonomy in Chapter 3.

1.2 Hierarchical Ensembles of Autonomous Decision Systems

The Hierarchical Ensembles of Autonomous Decision Systems (HEADS) framework is the foundation for bridging the gap between symbolic reasoning and data-driven approaches to system autonomy. Based on combinations of multiple expert systems, the framework attempts to reduce the decision making problem into many sub-problems that can be independently solved and combined into a robust system. Many of these sub-problem experts are combined to collectively solve singular problems, further enabling the separation of solution finding across many perspectives or domains of expertise. This approach is expert-agnostic meaning the sub-problems can be solved in the most appropriate manner (logic systems, filters, machine learning models, etc.) as long as the output includes a measure of confidence. This many-agent approach combines individual solution contributions to arrive at a cooperative ensemble output.

The primary intent behind this approach is to reduce total system complexity while maintaining broad and robust performance. The reduction of the decision making problem into sub-problems reduces the logic space by first considering if an input is relevant to the single output in question. If there is an input-output correlation for the sub-problem, then that input is added to the list of necessary inputs that a sub-problem agent can use. Ignoring uncorrelated input-output combinations for the sub-problem yields an exponential reduction of logic space. This serves to reduce the complexity of the sub-problem for both design and computation.

Aligned with the human-centric design focus of the problems of interest, this dissertation considers fuzzy systems as the core agents. Compared to probabilistic uncertainty describing *whether* an event occurs, fuzzy theory accounts for the uncertainty of *degree* of occurrence. This enables reasoning with overlap between rule antecedent activation within the logic system. Enabling such overlap is a powerful way to model nonlinear correlations between inputs and outputs and nuanced state transition boundaries.

An implementation of the HEADS framework requires first identifying the data model including inputs, outputs, and intermediate signals. Each output and intermediate signal is supported by at least one ensemble of expert agents. Here, the focus on decision making using fuzzy systems requires the If-Then rules to be developed next followed by the rule antecedent (fuzzy set) parameterizations. This bottom-up methodology enables the designer to focus on state-action mappings using the strength of linguistic variables inherent to fuzzy systems.

Finally, operational objectives are used as the scoring methodology to assess performance. The score is used to assess behavior across the system hierarchy and for guiding parameter optimization. Design metrics including signal space coverage and complexity measures assist the designer in assessing understandability and sensitivity.

1.3 Historical Foundations and Perspectives

Automatons are found in storytelling throughout history, but the practical development of autonomous systems has occurred over the last century starting with Alan Turing's first notions of artificial intelligence [5]. A rapid evolution occurred over the subsequent decades leading to autonomous system theories and implementations. A primary distinction must be made between automation and autonomy [6]. Automation relies on prescribed activity and feedback to map states to actions. Autonomy requires a more complex understanding of the self and motivation. Many competing autonomous agent architectures have been developed with the most capable incorporating planning, coordination, and reactive controls across multiple layered subsystems [7]. These complex autonomous systems have been enabled by generations of technological advancements in computing, sensors and actuators, and communications. Large collections of independent systems have been connected across networks further enhancing the distributed system's capabilities across physical distance and time [2, 8].

As systems become more complex through the combination of many component autonomous systems into large networks of distributed systems-of-systems, the challenges of verification and validation (V&V) of safety and performance become prohibitive to integrating such complex systems into human-centric applications [3, 9]. Approaches to solving this problem include modular V&V of individual subsystems and data-based modeling and simulation [10].

The recent proliferation of artificial intelligence (AI) and machine learning (ML) for modeling complex relationships based on empirical data has enabled high performance automated systems [11, 12, 13, 14, 15, 16]. However, these AI/ML models are difficult to generalize and adapt which makes them insufficient for system autonomy. Further, AI/ML approaches suffer from an explainability crisis comparable to V&V for complex systems [17]. The implementation of system autonomy in a manner that supports application to the NDSS problem while also supporting V&V is therefore a meaningful objective.

1.4 Difficulty in Praxis, or Limitations of an Engineering Dissertation

Generally speaking, when taking an idea from the theoretical to the practical, assertions about the validity of assumptions and real-world performance must be proven or at least rigorously supported. This holds true for most engineered systems, where theory informs practice and theoretical systems are modeled, prototyped, and tested before ultimately being produced at scale. However, the costs associated with rigorous validation and testing for human factors can be prohibitive. Instead, assertions concerning human comprehension can be assessed against and supported by the literature. In this dissertation, basic principles are discussed including information abstraction, understanding complexity, and cognitive loading. In the context of human factors and understandable system autonomy, these basic principles are explored through literature review. Future work in this area could include foundational human psychology and physiology to understand and explore the limits of cognition under realistic operational settings. Further exploration of human-in-the-loop interaction with the HEADS framework in simulation would be of great value for applications of the technology.

1.5 Dissertation Structure and Motivation

1.5.1 Chapter List

This chapter has covered the basic problem description, the contributions from this dissertation, and notes on the history of and current trends in agent based autonomy. The limitations of testing assertions about human factors and the mitigation strategy of literature-backed discussion employed here are acknowledged.

1.5.1.1 Chapters 2-4: Foundations

Chapter 2 presents a discussion of automation and autonomy, an overview of the history of autonomous agents, and recent trends in supporting technologies and requirements for human-centered systems. The problems arising from the concept of system complexity and how complexity impacts autonomous system design is covered in Chapter 3. The ideas of measuring performance and what 'truth' means for autonomous systems is also discussed. Chapter 4 includes practical approaches to modeling uncertainty and complexity and the important topic of how human factors are treated throughout the rest of this dissertation.

1.5.1.2 Chapters 5-7: HEADS Principles and Design

Chapter 5 presents the inspiration for and general approach to the HEADS framework. The attributes and structure of the framework are discussed in depth. Chapter 6 follows with the design methodology for implementing the HEADS framework and provides tools for designers to assess system completeness, complexity, and sensitivity. General performance is discussed in Chapter 7 including a standardized method for both defining goal-based performance metrics and using those metrics to enhance system performance through optimization.

1.5.1.3 Chapters 8-11: Practical Applications

The HEADS design methodology is applied to a representative NDSS problem in Chapter 8. The simulation environment developed for training and testing the implementation of HEADS is discussed in depth. Performance metrics are outlined and justified. The example HEADS system is presented in full including application of the introspective design metrics. Simulation results are presented with performance analysis for the original system design in Chapter 9. Tuned performance, first through manual analysis and tuning and then through application of evolutionary optimization to parameter modification, is shown in Chapter 10. Concluding remarks are in Chapter 11.

1.5.2 Alternative Reading Paths

A reader who is primarily interested in the HEADS framework design and implementation should first read Chapter 5 to understand the design choices behind the framework, followed by Chapter 6 for how to design a workable system. Optimizing for performance is found in Chapter 7, although for general cases the as-designed system should offer reasonable performance.

To understand the full sequence of technologies in the human-centered decision making framework of HEADS as used in this dissertation, start with Chapter 4 then proceed through Chapters 5-6 for implementation details. This path covers the basics of fuzzy systems, ensembles, and hierarchies and provides useful context for some of the assertions made about system understandability.

The reader who is already familiar with HEADS and who only wants a practical example should start with Chapter 8. Chapters 9-10 provide increasingly optimal performance results through initial design, manual tuning, and algorithmic tuning with narrative on how such improvements were achieved.

1.5.3 Motivation

This research was motivated out of a passion for exploring and developing space-related technologies that will benefit interplanetary human space travel. Based on the idea that the first crewed mission to travel beyond Earth orbit will be subject to minimal crew size and disruptive communications constraints with Earth-bound support, this framework contributes to the on-board system autonomy that will be necessary for safe and effective exploration.

The focus on human-centered engineering is founded on the belief that technology exists to serve human curiosity and that great technological advances cannot be devoid of human factors. Taking humans out of the equation not only reduces the anthropomorphic nature of a given technology, it also reduces the humanity of any person that must interact with or rely on such a system.

Finally, engineering is a cross-disciplinary art. It takes many experts from multiple domains to develop meaningful solutions to the hard problems faced by people every day. This is evident in this work through the multiple-agent based ensembles that were modeled after committees of distinct experts tasked with answering a single question. Each individual may not have much to say when faced with any specific problem and environment, but the team is better for its breadth and depth of focus.

Chapter 2: Foundations of Autonomy

The history of foundational advances toward practical system autonomy are many. The following (incomplete) list of contributions is offered as inspiration for the concepts discussed through the rest of this chapter:

- Francis Bacon developed the theory of inductive reasoning [18],
- Blaise Pascal invented the mechanical calculator [19],
- Charles Babbage invented the analytical engine [20],
- Ada Lovelace developed the first algorithm for implementation on a computer [21],
- Alan Turing pioneered artificial intelligence [5],
- John von Neumann and Oskar Morgenstern developed game theory [22], and
- Arthur Samuel created one of the first machine learning algorithms [23].

2.1 Automation and Autonomy

The separation and overlap between automation and autonomy is discussed in [6]. Automation includes prescribed activity and reactionary adaptation. The most straightforward examples of automation are scripted:

- a computer script follows instructions from start to finish
- a manufacturing machine takes feedstock and produces bent metal widgets through a series of operations controlled by mechanical cams and linkages

Automation can also include feedback-based adaptation:

- a computer script checks for return codes after executing each instruction and has logic to determine which of two actions to take based on the result of the previous step
- a manufacturing line includes robots that use motor encoders to measure end effector position and proximity sensors at the welding tips to control weld depth

Where automation is scripted and may include feedback control, autonomy requires more complex consideration of state and motivation. Autonomous systems do not exclude automation (in fact, many autonomous systems have high degrees of automation through component functions). The differentiating factor is how the system selects its next action. Automation relies on pre-defined sequences or control actions to drive system states toward desired values. The autonomous system considers multiple feasible actions against dynamic operating requirements and selects an action believed to achieve its goals. Selecting an action to achieve goals requires inherent understanding of the environmental state, the system state, how state evolution is correlated, and what actions are likely to result in desired effects. External motivation does not reduce autonomy, since an agent must still assess its state and environment to select a course of action to meet externally-defined goals.

2.2 Autonomous Agents

The discussion concerning agency as applied to computer systems is thoroughly treated in [7]. Briefly, there are three key issues in discussing agency, namely agent theories, agent architectures, and agent languages:

- Agent theorists are concerned with defining agents through formalisms and properties.
- Agent architects are concerned with implementing agents through structures.
- Programmers leverage agent languages to define agent programs to implement on computers.

A formal description of an agent for agent based reasoning, found in [24] as adopted from [7], is summarized as follows. The agent exists in an environment defined by states

$S = \{s_1, s_2, ...\}$ and can execute actions $A = \{a_1, a_2, ...\}$. The simplest definition of agent is a function

$$act : S \to A \tag{2.1}$$

which maps environmental states to actions. A more realistic agent first perceives and then acts

$$observe : S \to P \quad , \quad act : P \to A \tag{2.2}$$

where P is the set of percepts, and *action* now maps these percepts to actions. The agent may be more complex with many possible internal states K with internal state evolution driving actions:

$$observe : S \to P \quad , \quad orient : K \times P \to K \quad , \quad act : K \to A \tag{2.3}$$

2.2.1 Agent Theory

In [7], many possible properties of agents are described as attitudes and include belief, knowledge, desire, intention, obligation, and choice. An agent theory combines two or more of these attitudes and creates the formalism for how they represent the agent. The theory captures what an agent can do and by exception what it cannot do.

Agent theory matters for discussion of autonomous systems because every system has a set of attributes that the designers felt important enough to bestow on the system. These attributes define how the system perceives signals and uncertainty. The system exhibits the attributes through its logic. Defining these properties sets expectations for what the autonomous system is capable of and how it will behave.

2.2.2 Agent Architecture

The agent architecture incorporates the structure of modular components that interact to realize the theory. In [24], four basic agent architectures are described, including logic based agents, reactive agents, belief-desire-intent agents, and layered architectures.

2.2.2.1 Logic Based Architecture

A logic based agent considers the internal state of an agent and what it can perceive and provides mappings to feasible actions. The logic that provides these mappings incorporates both performance measures and uncertainty to support a system that evaluates actions based on known or estimated parameters. These feasible actions are provided to a planner function. A drawback of logic based approaches is the computational complexity of considering all feasible actions. To overcome this, a stochastic approach to uncertainty combined with performance measurement (cost, reward, fitness) has been demonstrated as practical for logic based decision making for rational agents [25].

2.2.2.2 Reactive Architecture

A practical approach to artificial intelligence is best summarized in [26] where the argument is made that an intelligent system must be grounded by physical connections to the world. Symbolic reasoning is insufficient as praxis since there must always be the physical connection through sensors and actuators. Further, symbolic representations are necessarily abstractions and may require assumptions or simplifications that reduce precision and accuracy. Stated succinctly in [26], "the key observation is that the world is its own best model... the trick is to sense it appropriately and often enough." The result of this work was the subsumption architecture [27] in which a combination of augmented finite state machines react to sensor data or messages from other machines to make control decisions. Subsumption describes a layered approach to decomposing problems into subproblems, solving the subproblems, and then composing solutions. The tightly-coupled sense-act paradigm leads to behavioral decision making to answer the question of what to do when, and how to do it. Behavioral actions are reactive in nature, where overall system behavior emerges from the interaction of individual behaviors.

Reactive agents are inherently similar to feedback controllers, since decision making depends only on the current perceived state. Reactive agents may require additional subfunctional connectivity or a higher level of determination before they can be considered to make decisions. Common methods for achieving this higher-level function include voting methods [28]. The speed of reactive agent architectures is often their most attractive feature. However, since no models are used, it is difficult to use non-local or estimated information as part of the decision making process. Deconfliction of competing priorities is also a challenge, since sub-functions have no inherent ability to consider their impact on achieving goals. One approach for solving this problem includes a blackboard model [29] for information sharing between reactive behaviors [30].

2.2.2.3 Belief-Desire-Intent Architecture

Perhaps the most common architecture endowing its agents with belief, desire, and intent is the procedural reasoning system [31]. Within this architecture, action selection depends on current goals, beliefs about the environment, and prior intent. An agent may also perform introspective reasoning about the validity of its beliefs, desires, and intent. External influences can interrupt or force an agent to reprioritize its goals. This architecture enables robust planning and plan adaptation in dynamic environments.

2.2.2.4 Layered Architecture

A layered architecture incorporates component functions of reactive, logical, behavioral, or belief-desire-intentive architectures across multiple independent subsystems. Early layered architectures are covered in depth in [32]. Applications of layered architectures may focus on the different layers, although they hold in common the separation of how (behaviors), when (planning), and what (connective courses of action between behaviors and plans). The general hybrid description of a layered architecture, with special attention to the reactive layer, is detailed in [28]. In [33] it is shown that a modular and hierarchical structure of behavior-oriented components enables complex reasoning.

2.2.3 Agent Language

Traditional software engineering approaches to modularity and hierarchy are well-suited to the design and implementation of an agent language. Throughout the literature one can find examples of agent languages that have become more historical than practical. This is attributed to the rise and popularity of powerful and flexible scripting languages (and abundantly available code examples) that make prototyping software simpler. The critical enabler of an agent language is the clear and concise definition of behaviors and attributes as outlined in the theory and structured according to the architecture. The reader is referred to [7] for further coverage of the topic.

2.3 Current Trends in Technologies Supporting Autonomy

Systems that exhibit autonomy depend on advanced technology to be able to perceive and interact with the physical world. Networks of autonomous systems multiply the importance of technological enhancement as more data is created, processed, and communicated. The following sections discuss a few critical technologies that support high-performance autonomous systems.

2.3.1 Technological Advancement

Advancement in technology drives enhanced capability of sensors, actuators, and computer devices even as the size, weight, and power (SWaP) of the devices decrease. Many common systems are locally-connected collections of these advanced devices including ground transportation vehicles, seafaring vessels, aircraft, spacecraft, smart buildings, and datacenters. These individual systems can do more than ever before across the land, sea, air, space, and cyber domains. The lower costs and increased capabilities of devices has led to the proliferation of independent and networked smart devices and systems. A distributed system of systems extends these devices across time, space, or both to deliver pervasive and persistent capabilities.

2.3.1.1 Advanced Computing

Moore's Law defines the trend that, since its origin in 1965, has empirically described the exponential enhancement of computing devices [1]. Central (CPU), graphics (GPU), tensor (TPU), and combined processing units support the SWaP reduction of modern devices. Not only are integrated circuits (ICs) becoming more dense, the increasing variety of application specific ICs (ASICs) means the performance enhancements are seen across general and specialized computing needs. An ASIC may be designed for signal processing, data acquisition, radio frequency digital filtering, or any other application that requires high performance and low latency. Field programmable gate arrays bridge the gap between ASICs and more generalized processors, with software-defined connections between logic blocks that can be reprogrammed after manufacturing.

The connection between advanced computing and autonomy is faster and more efficient processing. Autonomy for complex systems requires complex solutions which often require high-performance computer systems. Historical high-performance computers were not embeddable within the autonomous device. However, embedded computers available today offer performance comparable to historical supercomputers with minimal size, weight, power, and cost. Enhanced computers also benefit system designers through faster development of autonomous system technologies.

Another recent advancement in computing technology is the 'Cloud.' A large collection of networked computers serves as the backbone for nearly every consumer-facing or private internet-connected service today. Furthermore, many services imbued with facets of autonomy are backed by the Cloud, including voice recognition and networks of smart devices. While off-device autonomy is useful for some classes of problems, the inherent limitations of proximity to the datacenters that support such autonomy makes this approach infeasible for the class of problems considered in this dissertation.

2.3.1.2 Advanced Components

Advanced computing has further impacted the landscape of materials, peripheral devices, sensors, actuators, and other components commonly used within systems. Computer-aided research, design, and production of materials and widgets yields both advanced and application specific tools to solve problems more efficiently. Through embedded computer systems, intermediate process control can be accomplished at increased scale and efficiency.

The core devices and components used to build complex and capable systems directly enable system autonomy. Without such components an autonomous system would require more size, weight, or power to achieve the same effects. At the network level, advanced materials, sensors, and actuators enable more robust and resilient performance which helps reduce operating and maintenance costs. An autonomous system realizes its greatest potential when it is given the most advanced tools to interact with its environment.

2.3.1.3 Advanced Communications

The most common physical medium for wireless communication is radio frequency (RF) bandwidth. RF bandwidth is limited, as multiple users on the same channel interfere and render both communication links unusable. More efficient use of frequency spectrum through medium access control is one approach to solving this problem. Strategies to coordinate shared spectrum have been implemented such as frequency (FDMA), time (TDMA), code (CDMA), and orthogonal frequency division multiple access (OFDMA). Methods for deconfliction without pre-coordination, such as ALOHAnet [34], are based on a reactionary and stochastic approach. Advanced computing and components enable the use of higher frequency spectrum which provides more bandwidth leading to higher data rates at the cost of generally lower signal propagation.

Optical wireless communication is a newer capability that promises more throughput and less interference than RF communication. Laser based solutions are highly directive. Challenges to optical communications include background illumination and optical receiver sensitivity. It is also difficult to create an omnidirectional optical receiver making simultaneous optical links a challenge. Mitigations including optical filtering and miniaturized redundant laser sources and sensors have enabled practical prototype devices that address some of these challenges [35, 36, 37, 38].

Wired communication leverages electrical or optical signals over solid physical media. Wired connections often achieve high data rates with little susceptibility to or generation of interference. Networks made from wired connections are the preferred choice for local interconnectivity among subsystems and systems. However, wired networks are impractical for dynamic network topologies.

Distributed network-level autonomy relies on inter-agent communication for agents to understand each other. High throughput communication may not be necessary unless the network is made of heterogeneous agents with distributed sensing, reasoning, and actuation capabilities. In this case, a high bandwidth network is critical to system performance. Finally, for distributed systems subject to human oversight, the streaming of raw or processed data may be required to provide situational awareness or for auditing purposes.

2.3.1.4 Network Connectivity

Advanced computing, miniaturized components, and advanced communications serve as the foundation for enhanced connectivity of systems. Often referred to as the Internet of Things (IoT), these networks consists of sensors and processors that enable a variety of data-driven services including:

- smart home lights, locks, thermostats, and other devices,
- tracking and locating for postal and package delivery services,
- real-time information about transportation and logistics networks,
- identity authentication and authorization services, and
- management and control of distributed infrastructure such as electrical power distribution.

Although these networks often contain distributed systems-of-systems, their primary intent is more automation than autonomy [2, 8].

Connected networks of devices serve autonomous systems in a supporting role through their proliferation. The network of sensors, as a layer within an autonomous agent architecture, provides the autonomous agent with sufficient information for nuanced decision making.

2.3.2 The Need for Verification and Validation

Enhanced technology enables high-performance networks of autonomous systems. These systems must operate in a human context which requires strict validation and verification (V&V) driven by safety and regulatory considerations. Safety takes two forms:

- human safety and well-being when depending on or interacting with the autonomous system, and
- safety and longevity of the autonomous system itself.

The ethical nature of prioritizing safety is beyond the scope of this work. However, the basic requirement is that the autonomous system should do no harm to itself or others. Regulatory concerns are important when considering the application of autonomous systems outside of the laboratory where the impacts of failure may extend beyond the experimental system.

Increased system complexity makes validation and verification of systems more difficult [3, 9]. Research into methods for V&V of complex models has recently focused on leveraging tools such as model based systems engineering (MBSE) and systems modeling language (SysML) [10]. These tools are used to perform analysis of data flows, information consistency, and cause-effect interactions among subsystems during the design phase.

Operational considerations for V&V are focused on performance. One approach to guarantee performance is the separation of duties across hierarchical layers. For example, separating the decision and control loops can lead to guaranteed closed-loop tracking performance toward the dynamic outcomes from the decision making loop that often take more time to adapt and reconsider [39]. The separation approach enables each sub-part to be independently verified. However, independent verification may not consider the complex interactions between subsystems. To account for this complexity, enforcing a standard model for requesting and delivering data between subsystems can support inter-agent validation at the system level [40].

Another V&V task is understanding the bounds of system behaviors. The descriptive models that MBSE and SysML help develop can be used for on-line fault identification and tracing the impact of component failures across the system. Input validation is another method for rejecting out of bounds or corrupted data that might otherwise cause internal failures. The difficult task of full system characterization requires robust models and/or real hardware and sufficient representative data for simulating the range of anticipated operating environments and all possible fault conditions. Of course, testing at this scale and fidelity is either cost prohibitive or technically impossible on standard mission development timelines requiring high fidelity simulation and other approximate methods [41]. Therefore, the reliance on subsystem-level V&V and as much cross-functional testing as can be done in concert with simulation is the industry standard when it comes to integrated system V&V.

2.3.3 Artificial Intelligence and Machine Learning

A discussion on technologies and current trends supporting autonomous systems is not complete without mentioning artificial intelligence (AI) and machine learning (ML). At its core, artificial intelligence enables machines to execute tasks that would require intelligence if performed by humans [42]. AI is the broadest term for a system that incorporates functions of automation, reasoning, and autonomy. Machine learning is itself a broad subfield of AI that is focused on automated model generation and adaptation based on data processing and feedback [43].

The overlap between AI, ML, and autonomous systems means distinguishing between them is difficult. In practice, AI and ML are often conflated and used interchangeably. The proliferation of AI and ML solutions to complex problems is well documented across domains, including:

- AlphaGo is perhaps the most well-known artificial intelligence derived from machine learning and was developed to learn how to play Go [11, 12],

- medical diagnosis has been supported by machine learning models inspired by biology [13],
- machine learning has been used for model generation and optimization for electrical grid simulation resulting in faster high-fidelity simulations [14],
- spacecraft anomaly detection uses machine learning for categorizing complex multi-dimensional telemetry data [15], and
- asymmetric encryption methods are developed using generative adversarial networks [16].

However, most if not all current AI/ML models lack defining attributes of autonomy including belief, motivation, and the notion of *understanding* internal and environmental states.

Although highly informative when considering system autonomy, a deeper discussion on methods for designing and using AI/ML models is beyond the scope of this dissertation. The interested reader is encouraged to seek references on the topics of classifiers and various types of artificial neural networks (deep, recursive, generative adversarial, etc.) [44, 45, 46, 47, 48, 49] and tools useful for automated learning and model optimization (bagging, boosting, ensembles, evolutionary algorithms, etc.) [50, 51, 52, 53, 54].

2.3.3.1 AI/ML and Autonomy

State of the art AI and ML tools are adept at mapping knowledge of states into optimal categorizations or control decisions. When extended to searching a set of feasible actions the AI/ML approach can yield outstanding results exhibiting behavior adjacent to automated decision making. However, these tools are not malleable. Once a model is generated or trained it is effectively static. Performance can be generalized through training on a wide scope of plausible scenarios, but the first time an AI/ML model encounters an event that is not covered by its training set performance suffers. One approach to mitigate this effect is transfer learning, where a model is trained on representative data but applied to similar-but-not-exact data in practice [55, 56]. Still, to cover a scenario that has little or no overlap with the training scenarios still requires the development or training of a new model which can be a long and costly process. This makes standard AI/ML approaches insufficient for achieving system autonomy. Furthermore, the difficulty in verifying and validating AI/ML solutions for safety and bounded performance is an active area of research that has yet to bear meaningful fruit [17].

In spite of the inherent limitations of AI and ML they can be useful in the context of system autonomy. When considered as part of a layered approach to autonomy, AI/ML models show their strengths through reactive agency and function-specific efficiency.

2.4 Agent: A Working Definition

Without delving much further into philosophical or ontological discussion and for the purpose of this dissertation, a workable theory and architecture defines the agent as a system of functions to ingest information

$$observe : S \to P \tag{2.4}$$

which processes and reasons about the information driving internal state transitions

$$orient : K \times P \to K \tag{2.5}$$

develops a self-confidence measure C in its internal state based on its internal logic L

$$decide : K \times L \to C \tag{2.6}$$

and arrives at executable courses of action based on self-confidence

$$act : K \times C \to A \tag{2.7}$$

This system of functions operates concurrently and is modeled after John R. Boyd's Observe, Orient, Decide, Act (OODA) loop[1] where agility enables the agent to respond to dynamic environments to drive the system toward success.

The agent theory for this working definition includes attributes of belief, obligation, choice, self-confidence, and selecting a 'good-enough' option quickly. The theory is structured into a layered architecture with logic-based agents at the core.

[1] John R. Boyd was a United States Air Force fighter pilot and military strategist. He created the OODA loop to describe the cyclic decision process through which an individual or organization reacts to events. The theory provides a solid framework for the necessary functions of an agent [57].

Chapter 3: The Complex Nature of Autonomy

A great deal of attention is placed on the concept of complexity throughout this dissertation. It is therefore worthwhile to define and understand what complexity means in the context of systems, decision making, and autonomy.

Various reasonable definitions and characterizations of complexity are found in the literature, including:

- complexity arises from system attributes including nonlinearity, uncertainty, dimensionality, and structure [58],
- model complexity depends on the number of degrees of freedom [59],
- complex class boundaries cannot be modeled linearly [60], and
- complex means entangled such that the complex system cannot be decomposed while maintaining capability [61].

Other descriptive language that applies to complex systems includes: opaque, black box, partially-observable, hidden, hard to describe, difficult to reproduce, disorganized, and chaotic.

The rest of this chapter attempts to make sense of complexity as it applies to autonomy. Approaches to modeling complexity are discussed. Relevant to the problem considered by this dissertation, the networked and distributed systems-of-systems structure is described as hierarchical connections between complex subsystems. Finally, the difficult nature of truth for an autonomous system is considered.

3.1 Modeling Complexity

The difficulty in modeling complex systems comes from the need to capture holistic behavior while keeping the model itself tractable. The two main approaches to modeling include top-down and bottom-up, and each handles complexity in different ways.

A top-down approach treats the effect to be modeled as an outline or a sketch and iteratively incorporates more explicit detail until the model meets its purpose [62]. The added detail may arise from disaggregating a complex function into its constituent sub-functions, new functionality to handle edge cases, or simply identifying interacting sub-parts of a mechanism. Each iteration better describes the system as a whole through its parts. Ensuring each part meets but does not exceed its performance criteria ensures the overall model achieves its purpose without too much development cost. The benefits of this approach include minimum model complexity to achieve its purpose (consider a spherical cow...) and cost efficiency through designing to threshold metrics. A major drawback of top-down modeling is the cost associated with adapting the model to other purposes. Re-using its sub-parts is difficult due to the original focus on a specific use-case.

The bottom-up approach to modeling starts with first principles and develops added functionality through the combination of lower-level parts. Examples of component parts that are used to create a bottom-up model include differential equations and stochastic processes. The primary benefit of this approach is that a deep understanding of the basic mechanisms behind a behavior enable the designer to carefully control and adapt components that depend on the well-understood parts. Another major benefit is that once a fundamental process is understood and modeled it can be used as a building block in innumerable more complex models. A clear drawback to bottom-up modeling is the potential for high costs associated with understanding the basic principles and experimentally validating the low-level models.

Modeling a complex system often requires both top-down and bottom-up approaches. Where fundamental interactions among parts of the system are known or can be investigated deeply the bottom-up approach can reduce modeling errors. The top-down approach can help capture higher-level systemic properties and encourage strategic investment into deeper understanding of system behaviors. Whichever modeling approach is prioritized, the resulting model of a complex system will be made up of many component parts. These components describe the complex internal state evolution of the bulk system when exposed to inputs and perturbations.

3.2 Complex Agents as Complex System of Systems

The reader is referred back to Chapter 2 for how the following description of an agent was formed. For this dissertation, an agent is a system of functions to:

- ingest information,
- process and reason about the information,
- develop a self-confidence measure in its internal state based on its internal logic, and
- arrives at executable courses of action based on self-confidence.

This agent clearly consists of multiple component parts. The individual components do not need to be complex (ingesting information is a simple mechanism). The agent as a whole exhibits complex behavior based on how it uses the information it has to reason about itself and take actions to achieve its goals. Depending on the specific implementation of such an agent, it is easy to see how individual processes may be nonlinear and sensitive to nuanced differences in state realizations. When such complex components interact, the overall system is also complex.

3.3 Networked and Distributed Systems-of-Systems (NDSS)

A network of distributed systems-of-systems (NDSS) is the ultimate realization of a connected capability. Devices (or agents) within the NDSS communicate across the network to share information that can lead to increased overall system performance. Thoughtful design of the NDSS can lead to benefits beyond added physical or temporal reach of the desired capability. Building the network using many redundant nodes leads to robustness against individual node failure. Strategically placing, orienting, or connecting complementary devices or systems can boost NDSS performance. Layering network communications across multiple physical link methods protects against disruptions. These complementary systems provide more data, increase the value of collected data, or enhance the flow of information across the network.

However, the NDSS is not a panacea for perfect data, operations, robustness, or redundancy. Installing, connecting, calibrating, and maintaining large numbers of devices or systems can be costly. Large networks of connected agents can stress communications infrastructure. The data produced by independent sources can require additional preparation, correlation, or validation before it can be trusted or used. Real-time command and control of large numbers of devices can require vast teams of operators. The NDSS may be made of distributed heterogeneous sub-components across many layers of capabilities and effects. Complex interactions between these distant and disparate systems can require intense monitoring for system performance and stability.

One approach for reducing costs associated with implementation and operation of the NDSS is to enable each component system with on-board autonomy. Enabling agents in the network to choose their courses of action reduces the need for centralized monitoring and task management. Autonomous systems can be made robust against delayed or disrupted communication with the rest of the network. On-board autonomy also supports intelligent communication where objectives and results are shared rather than raw data. Individual autonomy enables smart and efficient operation of single agents. NDSS autonomy requires multiple agents to cooperate in fulfilling shared objectives. Cooperative autonomy without central coordination is the primary enabler of large NDSS. Cross-system autonomy is required for efficient optimization at the agent level [40]. Each agent should be trusted to handle its own autonomy while also participating in higher-level cross-system planning and optimization.

NDSS autonomy should not be disconnected from the human domain, however, for observability, verifiability, performance, and ethical reasons. Engineered systems are designed to solve human problems and should therefore be subject to human oversight. Human-in-the-loop operation is common for many existing systems for which autonomy is a goal, for example, autonomous ground transportation vehicles. Human cooperation is a necessity for other autonomous systems such as life support for astronauts in Earth orbit and beyond. When lives and human rights are at stake, it is dangerous and unethical to trust opaque autonomous systems that cannot be proven to operate in a safe and bounded manner. When consequences are less severe, a black-box autonomous system cannot be relied upon to behave deterministically.

Therefore, for reasons of safety, stability, efficiency, performance, and ethics, an autonomous network of distributed systems-of-systems should:

- incorporate and enable cooperation between any number of systems-of-systems;
- support verification and validation of deterministic behavior through inspection;
- support human-in-the-loop cooperation;
- enable per-agent operation in the absence of local or global connectivity or centralized control; and
- behave deterministically in uncertain or unbounded environments.

Meeting these criteria will result in a capable, adaptable, performant, understandable, and deterministic autonomous agent. Connecting such agents through direct communication or measured or estimated parameters is the foundation for localized cooperative clusters.

3.3.1 A Representative NDSS

The following example illustrates a representative network of distributed systems-of-systems and how per-node autonomy enhances the network. Applying autonomy across a network of a distributed spacecraft system (DSS) can reduce the logistical burdens inherent to large networks of cooperating nodes [63]. Backed by inter-satellite links (ISL), the DSS can coordinate plans, share measured data, bid on tasks, and cover for degraded nodes, increasing overall system effectiveness.

Data sharing impacts both the local node and the network as a whole. Each node, given data about its neighbors, might choose slightly different actions to balance its strengths against deficient neighbors. If satellite B is between A and C, and B has much more stored energy, it may opt to perform energy-intensive tasks for longer to make up for the reduced capacity from the other nodes. A and C may then decide to perform medium-energy tasks or save energy to cover for the next objective. The group adapts to optimize for current and future performance.

Leveraging the ISL for consistent cross-network logistics, the DSS can also respond to emerging or opportunistic events at speed. If one node detects a signal that meets mission criteria, for example a severe weather event forming over the ocean, it can share time and location data to the network. Other nodes can then consider repurposing their sensors to cover the event. The network can increase coverage, enable continuous tracking, and prioritize relevant data for downlink, all without direction from ground control.

Cross-DSS data sharing also benefits fault mitigation. A single node may compare its own performance against what is reported by its neighbors as a way to increase fault detection. Maybe node A has failing solar panels as the root cause of its low energy. If A consistently determines it can perform high-energy tasks for less time than B or C, it may self-select for additional diagnostics on-board or from ground systems and operators.

3.4 Autonomy and Truth

When it comes to autonomous systems that make decisions and take actions, being correct is a difficult thing to measure. There are often decisions that are known to be good or bad and these can be measured and tracked directly. However, when there is no feasible good decision, and similarly when there are many good decisions, the concept of correctness becomes unclear. If a desired course of action is known, whether the autonomous system takes the action is a binary classification. The corollary bad decisions or failure results are likewise easy to categorize. Binary failures can be iterated and may include:

- allowing a battery to be completely drained,
- taking an action that is contrary to a goal,
- causing a traffic collision, or
- misdiagnosing a patient.

These failures are events, and the failure truth is binary. Avoiding failures can be countably scored. When binary classification does not fully describe an event or a key performance indicator, a cost function can be applied:

- expending more battery power than necessary,
- driving too fast for the road conditions, or
- administering the wrong amount of medication to a patient.

These events occur to degrees and the indicator truth is fuzzy. It is therefore more useful to describe these indicators over time through the summation or integration of a cost function.

However, avoiding failure is only one way to be correct. For a stochastic system, correctness is binary. The probabilistic model predicts a future state with some uncertainty and in finite time that prediction can be evaluated as true or false. A decision making system, on the other hand, can have multiple feasibly correct actions from which to select. Choosing a single course of action requires comparing the feasible decisions. This drives the need for a cost/reward structure for measuring decision correctness. Measurable outcomes drive this cost/reward assignment since assigning scores to a decision is otherwise meaningless. For expensive predictions or comparisons, it may be necessary to select the first feasible action that meets pre-defined criteria. Thus, a decision making system can operate with degrees of correctness.

Chapter 4: Managing Complexity

Where Chapter 3 discusses what autonomous system complexity can mean and how it impacts system development and performance, this chapter covers approaches for managing such complexity. Complexity management is critical to system design for human factors and validation and verification considerations. How complexity is modeled and minimized directly impacts the explainability of system behaviors and performance. The following sections describe aspects of autonomous system complexity and approaches to understanding and modeling them in ways that encourage ease of use for human designers and co-operators. These factors and the complexity modeling and management decisions making up the

4.1 Probability and Uncertainty

How uncertainty is handled impacts how a system responds to unknown inputs. For example, a Markov process is a probabilistic system that evolves based on the current state and the likelihood of an event. Probability accounts for whether something happens. Fuzzy theory and fuzzy logic systems [64, 65] account for a different kind of uncertainty: one where any number of events can co-occur to varying degrees [66]. Therefore, the Fuzzy Inference System (FIS) serves as a strong basis for a logic system that deals with complexity. Input signals used for complex logic often have poorly-defined boundaries. The resulting decision outputs tend to overlap when the system moves from one state to another. Within the FIS, logic rules combine fuzzy set activations that capture overlapping signals and states to enable logic that incorporates this kind of uncertainty.

Probability theory encompasses the uncertainty of whether an event will occur. Statistical modeling serves as the foundation for making predictions. The sum of two fair dice is known to equal 7 with probability 1/6. On a finite time scale, whether an event occurs is a binary truth. Thus, a prediction of 7 for every future roll of the dice yields the best performance even though it will be wrong 5 of every 6 rolls. For a simple independent trial, this model makes mathematical sense. But, being objectively wrong 83% of the time may be unacceptable.

Understanding the world through the lens of probability enables an autonomous system to make rational decisions without perfect information. Statistical correlation can fill in knowledge gaps based on likelihood of events or states being true. When making decisions about the future, the rational decision maker leverages data and models. Assigning conditional probability to events lets a stochastic model represent complex system behavior. If a specific parameter or state is known (or thought to be known) the other parameters and states can be found using maximum likelihood estimation. Of course, having an accurate model of the system is necessary for such estimation. For a complex system, uncertainty in the component models tends to amplify overall uncertainty, which also means event occurrence uncertainty increases. And yet, whether an event occurs (or whether an event is likely to occur) still only captures one sense of uncertainty inherent in the real world.

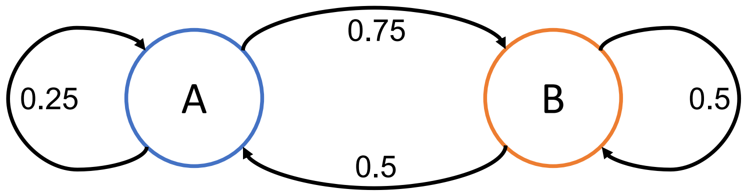


Figure 4.1 A simple Markov model

The nuance of *to what degree* an event occurs cannot be described under classical set theory. Binary set membership is inherently limited in the kinds of informational relations that can be drawn between state realizations. For example, Figure 4.1 shows a Markov process where the transition from state A to state B occurs with probability 3/4 and the B to A transition occurs with probability 1/2. This model captures that state B is likely to follow state A and that the system is most likely to be in state B at any time. What this model fails to represent is behavior at the state transition boundary. Further, this model assumes that state A is mutually exclusive of state B. Of course, a more complex model with intermediate states may better describe the state transition pathways and their likelihood, although the same inherent limitation of opaque transition boundary behavior and state exclusivity apply. For complex processes, transitions between states are rarely instantaneous or complete.

Under *fuzzy theory*, all things admit degrees [66]. Both probability theory and fuzzy theory describe uncertainty on the range $[0, 1]$. However, how the systems deal with a thing A and its complement A^c at the same time is the key distinguishing factor. Classical set theory requires mutually exclusive set membership. A stochastic event cannot have occurred and have yet to occur at the same time. A Markov model cannot be in state A and state B at the same time. Fuzzy theory allows for a more nuanced approach with degrees of set membership in multiple sets. Compared to the Markov model, a state realization where both A and B have nonzero membership activation levels is possible and commonplace under fuzzy theory.

The major contribution of fuzzy theory is the use of linguistic variables to describe signals, and fuzzy conditional statements and algorithms for manipulating linguistic variables. These features were revolutionary to managing the uncertainty of *to what degree* in the context of a human-comprehensible 'theory of possibility' [65]. It is the concepts of humanistic uncertainty and human comprehension of uncertainty that necessitate fuzzy theory for decision making.

4.2 Fuzzy Theory

Since Zadeh [64] first published on Fuzzy Logic as an extension of boolean logic that better captures uncertainty, many tutorials and applications have been published [67]. The Fuzzy Inference System (FIS) has been extended to be adaptive and combined with neural networks to increase performance [68, 69, 70, 71]. Fuzzy Logic is a non-probabilistic approach to addressing uncertainty which is concerned more with the *degree* to which something happens opposed to *whether* something happens.

4.2.1 Fuzzy Set Definition

A fuzzy set maps part of an input variable domain to $[0, 1]$ and is represented by a linguistic variable. An activation function (which can be continuous or discrete) and its parameters are chosen to enable sufficient coverage and overlap of the input variable domain into the fuzzy domain. Common functions include triangles and trapezoids due to their straightforward parameterization. However, any function that maps inputs to $[0, 1]$ can be used. There is no restriction on the dimensionality of the input space to a fuzzy set activation function. Similarly, functions that map non-real values to the appropriate domain may be used if needed.

Output fuzzy sets are similar to input fuzzy sets in definition and implementation. The main difference is that input fuzzy sets are used as rule antecedents and output fuzzy sets as rule consequents. The number of fuzzy sets used to represent an input or output can vary between fuzzy expert systems. For example, one expert might divide an input into three fuzzy sets, another expert might use seven fuzzy sets. This is a design choice driven by the granularity requirements of the rule set in use by the specific fuzzy expert. This variability is one way mechanism for increasing robustness, since an expert can be designed to perform well over a subsection of the input space. The first expert can then be augmented by another expert that is designed to cover a different input subspace.

4.2.2 Rule Definition and Evaluation

A rule is written in the form of standard logic using AND ($\wedge$), OR ($\vee$), NOT ($\neg$) and THEN ($\rightarrow$) terms and parenthetical grouping to evaluate antecedents into a rule consequent activation. For input fuzzy sets A, B, C, and D, and output fuzzy sets Y and Z, a rule set might include:

1. If A or C then Y
2. If not A or (B and D) then Z

Fuzzy logic rule evaluation combines fuzzy set activation levels for all antecedent terms. This process is referred to as 'fuzzy reasoning' as it generalizes the application of logic inference. The value resulting from fuzzy reasoning is the output fuzzy set, or rule consequent, activation.

Rule evaluation may apply to one or more antecedent terms. When multiple antecedents are considered, fuzzy logic operators dictate value propagation:

1. AND: (A and B) $\equiv \min(A, B)$
2. OR: (A or B) $\equiv \max(A, B)$
3. NOT: (not A) $\equiv (1 - A)$

Parenthetical statements group terms:

1. $(A = 0.5$ or $B = 0.8)$ and $C = 0.2 \rightarrow 0.2$
2. $A = 0.5$ or $(B = 0.8$ and $C = 0.2) \rightarrow 0.5$

The resulting activation level is then used to limit the output fuzzy set, or consequent, activation. This process is where fuzzy expert systems handle the uncertainty of 'to what degree,' as an output inference can be considered 'true' to the extent that the consequent activates. Figures 4.2 and 4.3 illustrate the evaluation of a multiple-antecedent rule and the consequent output fuzzy set activation that results.

4.2.3 Aggregation and Defuzzification

After rule evaluation, the fuzzy expert system consists of one or more output fuzzy set activations. Any particular output fuzzy set can have zero or more component activations that need to be combined via T-norm or T-conorm before the fuzzy expert system can provide a usable crisp output. The T-norm operator is a function

$$T : [0, 1] \times [0, 1] \rightarrow [0, 1] \tag{4.1}$$

that defines fuzzy intersection [72]. Common T-norms include the minimum and algebraic product. The T-conorm (or S-norm) similarly maps

$$S : [0,1] \times [0,1] \to [0,1] \tag{4.2}$$

Common S-norms include the maximum and algebraic sum

$$S(a,b) = \max(a,b) \tag{4.3}$$

$$S(a,b) = a + b - ab \tag{4.4}$$

The T-norm and T-conorm provide the basis for combining fuzzy sets via intersection and union, respectively.

The final duty of a fuzzy expert system is to produce a crisp output value built from the aggregated output fuzzy sets. This process is called 'defuzzification' as it takes fuzzy values as input and gives a crisp output. Common methods include:

- centroid of area,
- bisector of area,
- mean of maximum,
- smallest of maximum, and
- largest of maximum.

The computational burden of computing areas or means is one factor in deciding which defuzzification method to employ. The selection of a defuzzification operation may also depend on the output variable and whether the system must be lenient or conservative when reasoning.

Several fuzzy inference systems have been defined that provide the T-norm and T-conorm selections for both rule antecedent combination and rule consequent aggregation.

Some of these benefit from crisp outputs after consequent aggregation, reducing the computational burden from defuzzification. The Mamdani, Sugeno (TSK), and Tsukamoto fuzzy models are some of the most well-known and often-used throughout industry and literature [73, 74, 75].

4.2.4 A Fuzzy Example

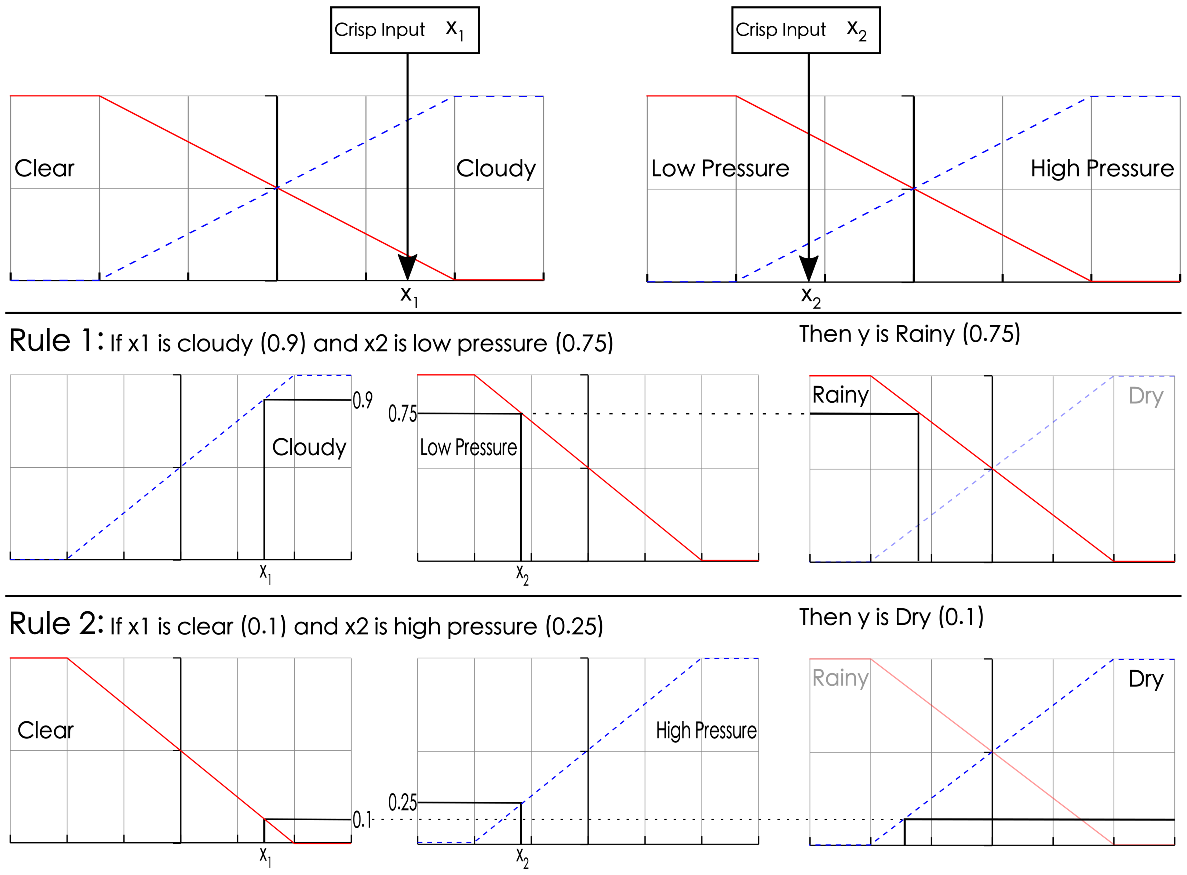


Figure 4.2 Fuzzy rule evaluation

Figure 4.2 shows an illustrative fuzzy logic calculation which maps two input signals, ‘pressure’ and ‘cloudiness,’ to a weather forecast, ‘rainy or dry.’ This example shows multiple fuzzy sets for both input and output, and the rules being evaluated are ‘IF cloudy AND low pressure THEN rainy’ and ‘IF clear AND high pressure THEN dry.’

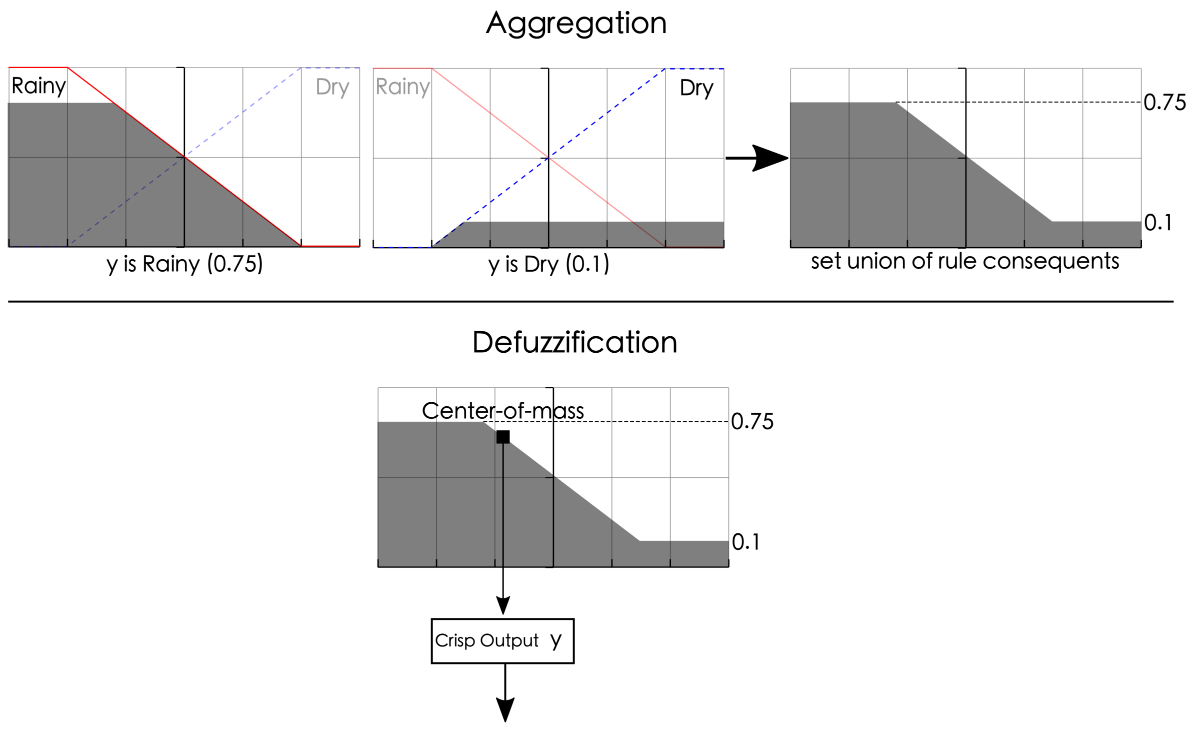


Figure 4.3 Fuzzy rule output aggregation and defuzzification

The aggregated output is the set combination 'rainy or dry' which is defuzzified by a center-of-mass calculation, shown in Figure 4.3. This simple example shows the power of fuzzy logic. Inputs yield multiple linguistic variables, which are inherently understandable, that when evaluated through the ruleset gives a crisp output backed by the output activation 'confidence' measurement.

4.3 Ensembles

A decision space is the number of logic rules needed to completely cover input-output combination mappings. Generally, the decision space dimensionality increases exponentially with an increase in the input space and linearly with outputs. It is impractical to design an autonomous system to cover all possible combinations. Without an enormous amount of data and precise knowledge of the operating environment, modeling the system is also impractical. Finally, failures and other perturbances will degrade the system performance in an unpredictable way. For a complex system, the list of unknown unknowns is impossible to investigate in full. Therefore, it makes sense to divide the problem into pieces that can be optimized for subsets of the solution space. When combined, such an ensemble can lead to robust and resilient performance.

Ensemble systems are a group of systems separately trained or otherwise managed that are tasked with addressing the same question. The outputs of all systems are combined to arrive at a single output. This provides the benefit of a variety of supporting systems that may provide an answer closer to the truth than any single expert could do. This 'Wisdom of the Crowd' concept is commonplace among machine learning systems today. The first efforts to increase the performance of neural networks using ensembles can be traced to Hansen and Salamon [49, 76]. The divide-and-conquer approach is also common to the blackboard paradigm that has seen wide application to multi-agent systems [77, 78, 30, 79].

With a sufficient number of individual systems trained across the problem domain, some are bound to show excellent performance and some will be poor performers. The goal is to train the ensemble such that the overall performance is acceptable across the range of anticipated operating environments or states. The output is either selected from the single best-performing subsystem or created by combining multiple outputs. The combination method is of interest, since even the best-trained group of experts cannot handle environments or scenarios for which it has no training overlap. However, an ensemble of subsystems may approach a reasonable solution due to partial overlap or adjacency between a new problem and the training data. Ensembles of systems have been proven to increase performance from classifier systems to control systems [44, 80].

Ensembles quickly evolved and many of the best-performing machine learning algorithms still rely on ensembles. One of the first examples of a mixture of experts ensemble regulated by a gating network was introduced in 1991 [81]. The general methodology behind an ensemble or mixture of experts is:

1. Train individual experts/systems on a subset of the operating domain, so that each expert is the most accurate on its domain;

2. Provide every expert with the same inputs;
3. Process the outputs to either:
    (a) Select the single-best performer and use its output, or
    (b) Aggregate the outputs from all experts.

The benefits of this approach include the ability to separate the input space across subsystems which reduces the overall computational complexity of the system (recall that the decision space for a classifier system increases exponentially with inputs and linearly with outputs). It is straightforward to negate relationships between inputs and outputs that are known to be uncorrelated. Reducing the input space for an expert to a subset of the overall input space may ignore input-output correlations. However, careful design can reduce the impact of such oversights and it is always feasible to add ensemble members to cover these missing pieces.

Ensemble structures have been used for fuzzy systems with success for classification, decision making, energy demand forecasting, and business strategy analysis [82, 83, 84, 85]. These applications leverage the strengths of fuzzy systems for modeling uncertainty. The ensemble structure enables sub-experts to be trained and combined which yields a more adaptable and robust system.

4.4 Hierarchies

Designing within a hierarchical structure is another tactic for dealing with system complexity. A hierarchical structure yields benefits ranging from varying control granularity, to covering uncertainty in system dynamics or environments, to multi-layered logic [86, 87, 88]. For systems consisting of multiple fuzzy logic experts the hierarchical structure fosters competition among experts that may be trained for different subsets of the input space. Compared to a flat structure, the hierarchy supports logical groupings that can lead to better understanding during design and increased performance during operation. Different levels in the hierarchy can also focus on various data including direct inputs, interpreted data, or other intermediate data. These intermediate levels in the hierarchy yield variables that can then be used as inputs by experts across the hierarchy.

There are myriad examples of hierarchical fuzzy systems. One of the earliest set out to reduce the complete rule base into a subset that includes only the most important rules [89]. Varying rule specificity across the hierarchy has been used to prevent less specific rules from diluting the value of the output [90, 91]. There are algorithmic approaches for decomposing a fuzzy system into two-input-single-output fuzzy logic units that leverage matrix representations [92]. Each of these approaches can result in increased performance using fewer resources during reasoning.

This structure can be extended into hierarchies of ensembles which are themselves hierarchical systems. Such an approach can further increase performance by taking multiple experts and connecting their mixed output to other experts in the system. Ensembles within and across layers can focus on decision making or control logic using the same input data. Separation of duties within the same hierarchical structure encourages logical system design which supports understandability.

4.5 Measuring Performance

The performance of a system can be measured through methods including cost-reward functions, binary measures of task completion, and error compared to a desired signal state. A classification system, for example, requires labeled data to assess performance of the classifier via measures including accuracy, precision, recall, and an 'F-measure' based on true-positive (TP), false-positive (FP), true-negative (TN), and false negative (FN) counts:

$$accuracy = \frac{TP + TN}{TP + FN + FP + TN} \tag{4.5}$$

$$precision = \frac{TP}{TP + FP} \tag{4.6}$$

$$recall = \frac{TP}{TP + FN} \tag{4.7}$$

$$F - measure = 2 * \left(\frac{precision * recall}{precision + recall} \right) \tag{4.8}$$

These measures provide a numerical score for system classification performance that can be used to objectively compare different classifier or other binary systems. There are many other metrics used in statistical analysis of information theory that are derivative of sensitivity (true-positive rate) and specificity (true-negative rate) as first described in [93]. For example, the receiver operating characteristic ratio of TP/FP can be used for assessing performance between different models or parameterizations to aid in selecting an optimal system [94].

When the system outputs are control decisions, cost-reward or error measures tend to provide a better assessment of performance. In cases where the truth or a desired state is known, performance increases as accumulated (counted or integrated) absolute error decreases. When truth or the desired state is unknown, scores are assigned to operating parameters and counted or integrated. For example, the general optimal control problem seeks to minimize cost (J) associated with the endpoint (E) and runtime (F):

$$J(x(\cdot), u(\cdot), t_0, t_f) = E(x(t_0), t_0, x(t_f), t_f) + \int_{t_0}^{t_f} F(x(t), u(t), t)dt \tag{4.9}$$

This equation considers cost for state and control inputs that may be nonlinear. The cost function for an autonomous system may be as simple as a counter index for positive outcomes. It is up to the designer to select and apply scoring in an appropriate manner since the chosen scoring methodology impacts performance assessment and optimization.

4.6 Notes on Computational Complexity

It is important to consider how effective a complex autonomous system can be when applied on low-cost and low-power computers. Technological advances have made these modest computers relatively powerful compared to historical devices. However, the autonomous system may share the computer with other applications and therefore must minimize its resource use. The following illustrates how ensembles, hierarchies, and subdividing the input space among multiple expert systems benefits computational complexity.

A complete logic space includes rules based on all possible combinations of inputs and outputs. The logic space dimensionality increases linearly with added outputs m and exponentially with added inputs n:

$$\#rules = m * (2^n - 1) \tag{4.10}$$

Intermediate variables as defined throughout the hierarchy can act as both outputs and inputs which further adds to overall complexity. Note that the null input is ignored which leads to the -1 adjustment to the exponential.

At the most basic level, breaking the logic space into output-driven subsystems that are fully-combined leads to each individual subsystem having complexity of $1/m$ compared to the single fully-connected system. A more profound reduction in complexity is seen when subsets of the input space are used to reason about a single output. Reducing the total number of inputs by q reduces the computational complexity by a factor of about 2^q. Even in the extreme case where all input-output correlations are used for the reduced-input sub-space, the complexity of the resulting system is a minor fraction of the complete system for relatively small q:

$$\#rules(reduced) = 1 * (2^r - 1) \tag{4.11}$$

with the reduced complexity ratio $\psi_{r/n}$ being

$$\psi_{r/n} = \frac{2^r - 1}{m * (2^n - 1)} \tag{4.12}$$

and the reduction in complexity as a percentage

$$\%reduction = 1 - \psi_{r/n} \tag{4.13}$$

Figure 4.4 shows how small reductions in the input sub-space can greatly reduce the total system complexity. The effect is more pronounced as the size of the input space increases.

For example, for $n = 30$ a reduction of only four ($r = 26$) reduces the sub-system complexity by 94% compared to the fully-connected system. Of course, multiple reduced-complexity systems are needed to provide adequate coverage and performance. The most effective balance of overall reduction and combination of reduced systems is left to the designer.

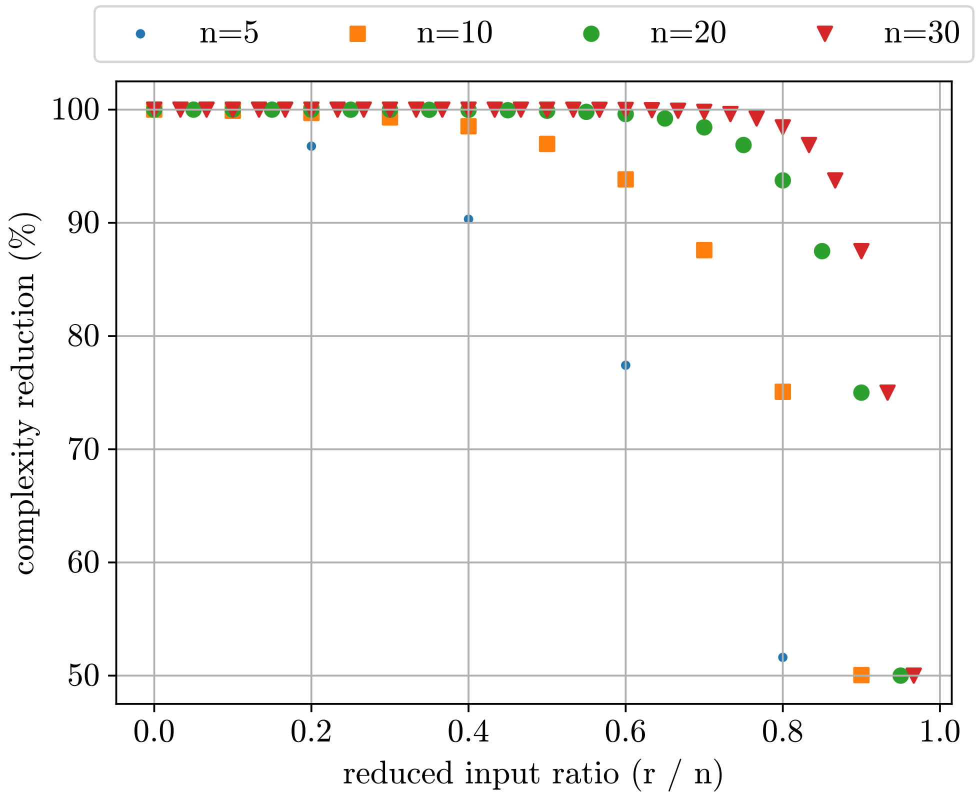


Figure 4.4 Complexity reduction as a function of reduced input sub-space fraction for various input space sizes (constant output space size of $m = 1$)

Considering multiple outputs amplifies the effect of system complexity reduction. Figure 4.5 shows this effect for a system with input space size $n = 30$ and output space size varied $m \in \{1, 2, 5, 10\}$. For system $m = 10$ a reduction of input space by one $r = 29$ results in a 95% reduction in sub-system complexity assuming the reduced subsystem is required to provide a single output.

This approach to assessing computational complexity accounts for the general number of operations required to process a system of arbitrary size. It is meant to illustrate the power of considering combinations of reduced subsystems compared to a naïve fully-connected

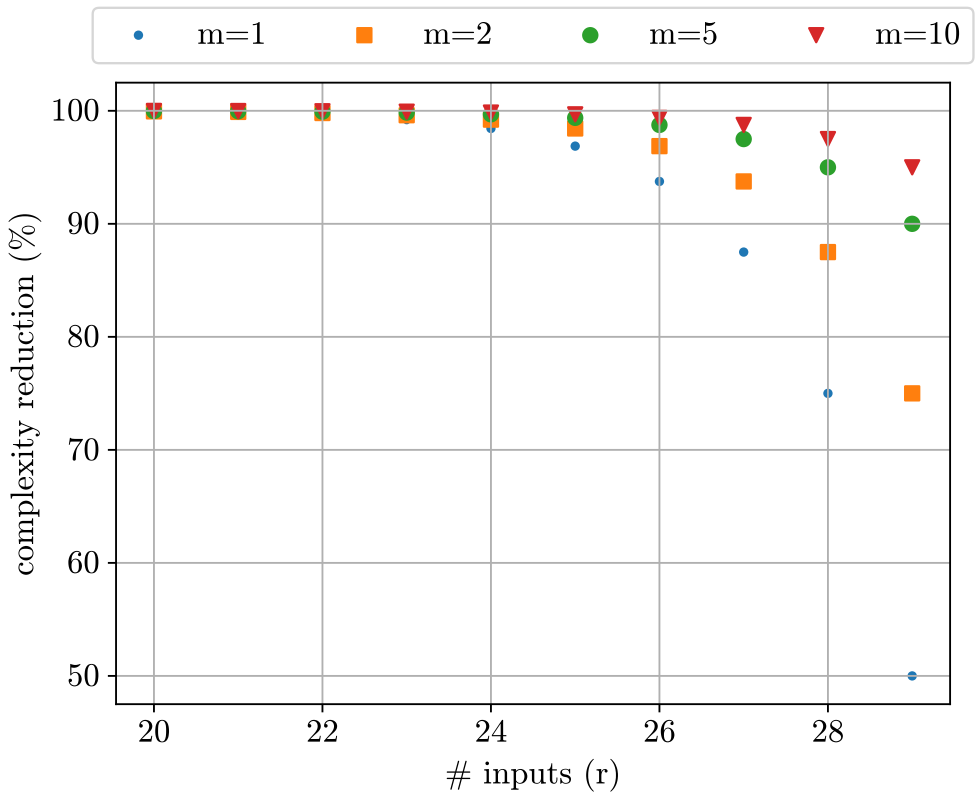


Figure 4.5 Complexity reduction as a function of reduced input sub-space fraction for various output space sizes (constant input space size of $n = 30$)

system. The system designer must carefully select signal subspaces since complex systems often have nuanced input-output correlations.

4.7 Notes on Human Factors

This dissertation makes broad claims about the importance of human understanding of autonomous systems. The scientifically rigorous method to prove these claims through testing research subjects' situational awareness while interacting with a simulated system would require expertise in assessing human cognition. While attempts have been made to make this approach easier by clearly outlining data collection strategies, data analysis techniques, and recommendations for selecting a strategy and technique [95], this level of effort is reserved for future work. The importance of human-machine interaction is well-established and studied [4, 96, 97, 98, 99, 100, 101, 102]. This section provides arguments behind the assertion that human understanding is fundamentally enabled through language and an overall reduction in complexity.

4.7.1 Language Is Human

Humans have a rich language capability that extends to the powerful cognitive capability of recognition and reasoning with language [103]. Human language conveys complex ideas, emotions, context, and other direct and indirect information. It is a tool for storytelling, describing, convincing, and learning. These attributes and functions make human language a strong foundation on which to build a tool meant for autonomy.

Humans can play many roles when interacting with an autonomous system. A fully-autonomous system might rely on humans for oversight. The system may provide data or other services to humans as customers. A cooperative autonomous system might augment or amplify human capabilities. A co-dependent human-machine system might rely on human-sourced information to give context to constrain the autonomous agent.

The difficulty in using human language as a basis for automation comes from the necessary translation into the computer domain. Encoding complex ideas conveyed through words or phrases requires a translation into a format through which a computer can process the data using models to infer meaning. Much work has been done toward giving computers the capability of natural language processing (NLP) [104, 105, 106]. Several applications of NLP for processing textual data include information retrieval (finding the most relevant document based on search query), information extraction, text classification, and text generation [107, 108]. These algorithms have shown good performance at their tasks. Training NLP models to understand emotion, nuance, or other complex characteristics of language is an active area of research [109, 110, 111, 112]. Understanding multidimensional contextual cues including auditory (cadence, inflection) and physical (facial expression, body language) is another area of research supporting human-machine interaction [113, 114, 115, 116, 117]. These challenges and their solutions combine toward computers generally understanding human input better. However, until machines can easily understand human communication their reasoning is limited to the vocabularies and ontologies granted by human designers.

Language-to-data abstraction through NLP models helps computers understand humans. Of equal importance is a natural language representation of computer-generated data in support of explainability and human comprehension [6]. In the context of explaining autonomous decision making, reconnecting the abstracted cause-effect relationships into 'why-because' statements supports understanding the system's beliefs and intents. Enabling the computer to reason using language-based data can therefore directly support human comprehension of such reasoning.

4.7.2 Linguistic Variables, Complexity, and Understandability

In defining the need for fuzzy theory, Zadeh contends, "the conventional quantitative techniques of system analysis are intrinsically unsuited for dealing with humanistic systems or, for that matter, any system whose complexity is comparable to that of humanistic systems" [64]. The linguistic nature of fuzzy set definitions naturally lends itself to human comprehension. A person can comprehend what a 'comfortable temperature' might be without seeing a number. Individuals might not each have the same numerical 'comfortable temperature,' but they can each understand logic based on such a value.

Fuzzy systems depend on linguistic variables for logic design. System designers write logic rules using natural language to describe relationships between data and also the variables that represent the data. This reduces cognitive loading on the system designer by capturing human-centered reasoning without numerical abstraction. Since they rely on linguistic variables, fuzzy systems support direct inspection during operation without the need to reconnect numerical data to meaning. Linguistic variables contain meaning once their fuzzy set activation functions are evaluated based on input numerical data. An activated variable can then be traced through rule evaluation to understand how and why specific rules activated. It is therefore straightforward to understand that the 'turn heater on' rule activated because the 'user cold' variable activated first.

Linguistic variables are the mechanism supporting human comprehension of fuzzy systems. However, complex rule structures and having many or poorly-differentiated linguistic variables can reduce understanding. It is useful to define metrics for complexity and understandability through which fuzzy systems can be compared. Completeness, consistency, and complexity measures for fuzzy logic rule understandability are described in [69]:

- completeness: for an arbitrary system input state $\mathbf{x}_0$ there must be at least one fuzzy rule that triggers
- consistency: a rigid definition requires that no rules can have the same antecedents but different consequents (although relaxation may be required with overlap between rules with multiple antecedents)
- complexity: a system with fewer rules is less complex

Notions of interpretability and readability are founded on minimizing the number of rules, the number of antecedents used by each rule, and total number of linguistic labels in use [118]. Low-level and high-level interpretability are distinguished as applying to fuzzy sets and fuzzy rules, respectively, and depend on factors including distinguishability, readability, and transparency of structure [59]. An interpretability index that integrates a membership function coverage measure and two separate complexity measures is proposed in [119]. A methodology for designing highly-interpretable linguistic knowledge bases measured by total number of rules, total number of premises, the number of rules using three or fewer premises, and the total number of labels per input is found in [120].

A common feature of these metrics for understandability is the minimization of the total size and complexity of the system. Reducing the layers of abstraction between inputs and their impacts on the outputs supports human comprehension. The linguistic nature of logic within fuzzy systems supports the best of both human and machine languages. Humans write and read logic rules using linguistic variables that represent subsets of input spaces. Computers use fuzzy set activation functions to obtain numerical representations of these linguistic variables for performing comparison operations. The fuzzy activation functions are the only abstraction layer between human and machine domains.

Chapter 5: Hierarchical Ensembles of Autonomous Decision Systems

The case for autonomy applied at the per-system and cross-system level for both local and connected distributed systems is evident. The tools and techniques for understanding and managing complexity described in Chapter 3 and Chapter 4 are the foundation that enables modeling and implementation of autonomous complex decision making. The resulting framework consists of hierarchical ensembles of autonomous decision systems (HEADS).

5.1 Principles and Attributes

Several overarching principles were considered when designing the HEADS framework for automated reasoning:

- a focus on human-centered design,
- requiring bounded behavior to support validation and verification, and
- an extensible design supporting simple modification after initial development.

These principles attempt to capture the critical attributes required for explainable autonomy for mission-critical applications and human-centric or human-adjacent operation.

5.1.1 Human-Centered Design

Autonomy can and often should be able to perform without human oversight for reasons of efficiency, speed, and cost-effectiveness. However, even the best fully-autonomous systems cannot match human capability across a variety of tasks. It has also been shown that human-machine teaming increases system performance even when the human input is only a third as accurate as the automated system [121]. A primary reason for this cooperative benefit is the type of data each uses for reasoning where a computer can operate on large volumes of labeled data while a human processes narrative and context with ease.

The HEADS design was modeled after collective human decision making embodied by NASA Mission Control where the hierarchy of decision making supports the management of some of the most complex systems ever designed. Mission Control spans automated control systems, teams of engineers dedicated to maintaining and evaluating data for each subsystem, consoles operated by subsystem-specific subject matter experts, and the Flight Director who serves as overwatch across subsystems. The HEADS framework structure leverages hierarchical ensembles to balance system performance and complexity. Fuzzy experts are the core logic expert systems and exhibit desirable characteristics including linguistic variable based reasoning and bounded outputs resulting in understandable and verifiable performance. Layered collections of cooperating experts structure enables robust multiple-objective decision making. The foundation built on fuzzy logic and linguistic variables enables a human-centered approach to modeling complex behaviors. The HEADS framework yields autonomous systems with inspiration from how humans consider information and perform reasoning.

5.1.2 Bounded Behavior

Another powerful attribute of the HEADS framework design based on fuzzy logic experts is bounded behavior. Through the nature of fuzzy systems design, unbounded input domains are subdivided into fuzzy set domains. Each fuzzy set domain can be bounded or unbounded but its activation is mapped to $[0, 1]$ by definition. Then, the fuzzy logic rule evaluation uses these activation signals as antecedents and yields yet another bounded output signal. When the output fuzzy sets are defined on a finite domain, such as $[0, 1]$ as done for logic consequents, the entire system yields bounded outputs.

Automated controllers, guidance algorithms, and even humans may rely on these bounded autonomous logic outputs as inputs. The most direct approach to using the logic outputs is to set an action threshold where the action is taken once the logic output meets or exceeds the threshold. Other methods might use linear or nonlinear functions to map the output in $[0, 1]$ to other values that are meaningful for the system. As a machine-machine translation or explainability aid, intermediate mapping functions can be used to extend the logic output ($[0,1]$) into extended-domain ($[a,b]$) data to better match expected values. Regardless of how it is used the logic output is verifiably bounded at all times and can therefore be mapped in a finite way to support validation and verification of system outputs.

5.1.3 System Modularity

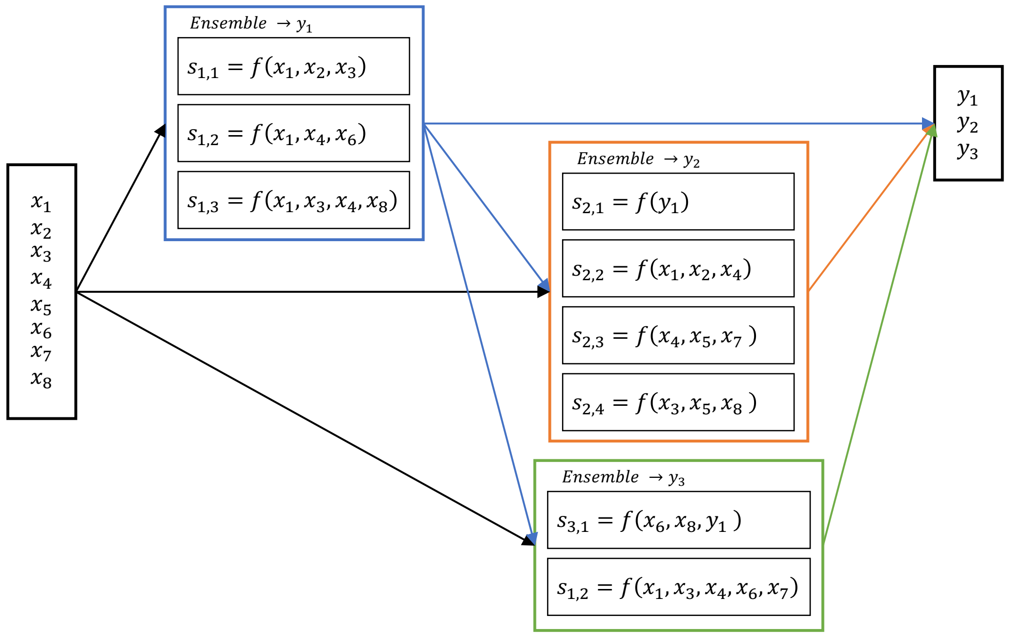
The modularity of the HEADS framework comes from the shared signal space and the divide-and-conquer approach to problem solving. The shared signal space is a pool of data containing system inputs and logic outputs. These signals can all be used for reasoning by one or more logic systems. A system designer can add ensemble subsystems at any time to provide new outputs to the signal space that can be used for reasoning across the rest of the system. Adding new experts within existing ensembles to use the new subsystem outputs does not require modification or retraining of the existing experts. The HEADS framework system can therefore accept new information with zero rework of existing subsystems. These extensions increase problem-specific performance and are effectively plug-and-play with the rest of the framework.

5.2 Structure

The following sections describe the HEADS structure. The signal space is formalized as the combination of all available inputs and outputs. A description of a fuzzy expert and its component parts is provided in the context of the signal space. The collection of experts in an ensemble and the interconnections between ensembles into the hierarchy complete the framework structure. Figure 5.1 shows the basic HEADS topology of multi-layered ensembles containing many expert systems that use subsets of the signal space for reasoning.

5.2.1 Signals

The signal space contains all inputs, intermediate outputs, and terminal outputs of the system.



$Ensemble \rightarrow y_1$
$s_{1,1} = f(x_1, x_2, x_3)$
$s_{1,2} = f(x_1, x_4, x_6)$
$s_{1,3} = f(x_1, x_3, x_4, x_8)$
y_1
y_2
y_3
$Ensemble \rightarrow y_2$
$s_{2,1} = f(y_1)$
$s_{2,2} = f(x_1, x_2, x_4)$
$s_{2,3} = f(x_4, x_5, x_7)$
$s_{2,4} = f(x_3, x_5, x_8)$
x_1
x_2
x_3
x_4
x_5
x_6
x_7
x_8
$Ensemble \rightarrow y_3$
$s_{3,1} = f(x_6, x_8, y_1)$
$s_{1,2} = f(x_1, x_3, x_4, x_6, x_7)$

Figure 5.1 Hierarchical Ensembles of Autonomous Decision Systems (HEADS) topology

The input vector includes data that are not generated within the system:

$$\mathbf{x} = \{x_i : i = 1, 2, ..., n\} \tag{5.1}$$

Intermediate and terminal logic outputs are treated the same and make up the output vector:

$$\mathbf{y} = \{y_j : j = 1, 2, ..., m\} \tag{5.2}$$

The combination of inputs and outputs is the full signal space

$$\mathbf{Z} = \{\mathbf{x}, \mathbf{y}\} \tag{5.3}$$

such that every signal $z_k, k = 1, 2, ..., m + n$ is information available for logic.

The use of fuzzy systems within the HEADS framework means the signal space incorporates both quantitative and qualitative data. Quantitative data includes measured, estimated, and derived numerical data:

- measured, e.g., battery voltage, storage space remaining, system temperature,
- estimated, e.g., physical state, sensor bias, duration until critical event, and
- derived, e.g., current power draw as a function of voltage and current, time since the last signal update or change.

Qualitative data leverage linguistic variables directly and are captured through binary or fuzzy set activation values. Qualitative data are interesting in the context of human-machine cooperation as they give the human a natural language format for providing the autonomous system information and feedback. For example, a logic system might reason based on human operator emotional state and rely on the human to provide signals such as 'happy,' 'sad,' 'stressed,' and 'anxious.'

5.2.2 Ensembles

A core feature of the HEADS framework is the collection of many expert systems into an ensemble. This enables decision making among a variety of independent experts that provide robustness in a dynamic problem space. The many experts deliver their outputs to the ensemble which collects, weights, and combines the independent outputs into a single ensemble output. Each ensemble output serves as an intermediate or terminal system output y_j. Multiple ensembles can be incorporated into an ensemble structure which itself provides a single system output.

Multiple experts solving the same problem leads to collaborative-competitive decision making where each individual expert impacts the ensemble output based on a confidence factor. HEADS leverages information from the fuzzy aggregation and defuzzification process as a measure of uncertainty in an expert's output:

1. each rule that activates limits its output fuzzy set to the rule activation level,
2. all such limited output fuzzy sets are combined according to the system aggregation algorithm, and

3. the aggregated output fuzzy set activation level that corresponds to the defuzzification output value is used as the uncertainty metric.

The output confidence is joined with the defuzzified output into a tuple of (value, confidence). All expert output tuples are combined at the ensemble level using the softmax function, a normalized exponential which generalizes the logistic to multiple inputs [122]:

$$O_j = \frac{\exp(I_j)}{\sum_k \exp(I_k)} \tag{5.4}$$

The softmax gives larger weight to relatively larger values in a set. Since the softmax is a function that maps real-valued inputs to $[0, 1]$, the outputs are often compared to discrete probabilities. The confidence values from the defuzzified expert outputs are the inputs to the softmax function, which then produces a weight for each expert's output value. The ensemble output is then the weighted sum of these softmax-computed weights and the raw output from the expert system.

A single expert that has a much larger confidence in its output will impact the ensemble to a greater extent. If all expert systems have similar confidence, then they will each impact the ensemble in a more even manner. The output confidence selection method used for ensemble mixing leverages the expertise for a realized input subspace. A confident expert activates when its rules and fuzzy sets align with the system state. The rule outputs for an expert aligned to the system state will aggregate around a narrower section of the output domain. The resulting crisp output value will effectively be known with higher confidence.

5.2.3 Expert Systems

Fuzzy systems are the core logic experts within the HEADS framework. Natural language logic modeling supports human comprehension through design and operation of the systems. Each individual fuzzy logic expert contains fuzzy sets and fuzzy rules that map inputs to an output through aggregation and defuzzification functions.

For each system output y_j there is a set of expert systems $\mathbf{s}_j$ containing p_j individual experts:

$$\mathbf{s}_j = \{s_{j,p} : j = 1, 2, ..., m; p = 1, 2, ..., p_j\} \tag{5.5}$$

To reduce both logical and computational complexity, an individual expert system within the ensemble uses a subset of the signal space

$$\mathbf{z}_{j,p} = \{z_k \subseteq \mathbf{Z} : z_k \text{ used by } s_{j,p}\} \tag{5.6}$$

The system contains fuzzy sets as functions of signals. Each individual expert system has its own collection of these fuzzy sets. The input subspace for a given fuzzy expert system is the full space for the fuzzy expert system's fuzzy sets:

$$\mathbf{m}_{j,p} = \{\mu_{j,p,u} (\mathbf{z}_{j,p}) : u = 1, 2, ..., u_{j,p}\} \tag{5.7}$$

where $u_{j,p}$ is the number of fuzzy sets in use by expert system $s_{j,p}$

An individual fuzzy set is defined for one signal within the subspace and maps the signal to an activation level:

$$\mu_{j,p,u} (z_{j,p,k} \in \mathbf{z}_{j,p}) \rightarrow [0, 1] \tag{5.8}$$

Each fuzzy set subdivides an input signal into multiple sections each described using linguistic variables. If the input is *temperature* the fuzzy set variables might include *very_cold*, *cold*, *warm*, *hot*, and *very_hot* with a single value of the input mapping to one or more fuzzy set activations.

There are also fuzzy sets for rule consequents, with one or more defined for a fuzzy expert system such that an output value yields an activation level:

$$\nu_{j,p,v} (y_j) \rightarrow [0, 1] ; v = 1, 2, ..., v_{j,p} \tag{5.9}$$

During processing output fuzzy activation levels are combined to produce the output y_j as part of the rule aggregation and defuzzification algorithm. The output fuzzy sets logically map to actions or decisions that the HEADS system is tasked with making. For example, an action might be *charge_battery* to determine when to prioritize lower power use or higher generative capacity, with component fuzzy sets including *charge_later*, *charge_soon*, and *charge_now*.

The fuzzy expert system contains rules that define the logic used for autonomy:

$$\mathbf{r}_{j,p} = \{r_{j,p,w} : w = 1, 2, ..., w_{j,p}\} \tag{5.10}$$

Each rule uses one or more fuzzy set to determine an output fuzzy set activation:

$$r_{j,p,w}\left(\mathbf{m}_{j,p}\right) \to \nu_{j,p,v} \in [0, 1] \tag{5.11}$$

Rules are written using standard logic operators (AND, OR, NOT) and parenthetical groupings (A OR (B AND NOT C)).

In processing the logic rules each fuzzy output set attains some activation level. These output levels are combined into an expert output fuzzy set $\nu_{j,p}$ according to one of multiple aggregation methods including the minimum, algebraic product, maximum, or algebraic sum. The default aggregation function used within HEADS is the maximum:

$$\nu_{j,p}\left(y_j\right) = \max_{v}\left[\nu_{j,p,v}\left(y_j\right)\right] \tag{5.12}$$

Once the component output fuzzy sets are aggregated the output fuzzy set $\nu_{j,p}$ is defuzzified using one of several common methods including centroid of area, bisector of area, and mean of maximum. The default method used in HEADS is the bisector of area:

$$y_{j,p} = t : \int_{-\infty}^{t} \nu_{j,p}\left(y_j\right) dy_j = \int_{t}^{\infty} \nu_{j,p}\left(y_j\right) dy_j = \frac{1}{2} \tag{5.13}$$

A feature of how HEADS treats this defuzzification is that it captures both the defuzzified output and the aggregated fuzzy set activation level of that output as its confidence:

$$c_{j,p} = \nu j, p\,(t) \tag{5.14}$$

The output and confidence values enable ensemble-level combination.

5.2.4 Hierarchies

Hierarchies arise from the using intermediate system outputs as signals for performing logic. These intermediate signals are defined in the output vector and as part of the signal space but are not required at the system edge for data egress. The utility of such intermediate signals is aligned with the subsystem nature of the HEADS framework leading to components of lower complexity that combine to enable complex behavior. In the context of using fuzzy systems as experts, these intermediate signals can be quantitative or qualitative in nature leading to both computationally and linguistically powerful subdivided logic. The hierarchical structure is not formally defined but rather an emergent property of the data flow between ensembles.

5.3 HEADS as an Adaptable Framework

The definitive purpose of HEADS is to describe interconnectedness and output data mixing techniques to combine many independent expert systems to achieve autonomous decision making for complex systems. Specific choices have been made thus far to maximize the framework for human-centered design and interaction. The framework could be adapted in various ways to meet other requirements or constraints. This section discusses specific features of the HEADS framework and implementation as presented and alternative choices that could be made.

5.3.1 Expert System Selection

Fuzzy systems were selected as the core expert system to provide HEADS with the direct benefits of linguistic variables including understandability without translation. An additional feature of fuzzy systems is how they model uncertainty similar to how humans manage uncertainty in decision making. These features support the need for human understandability and oversight for autonomous systems that operate adjacent to people.

Other expert systems may be appropriate depending on understandability and performance requirements. Any model or system that provides output with some measure of confidence or uncertainty can be a drop-in replacement to an ensemble. Experts of various types can be combined in an ensemble under the same constraints of requiring output data with confidence metrics to be used for output weighting.

5.3.2 Heterogeneous Distribution of Experts

A naïve approach to system design would be to create an equal number of experts within each ensemble. There is no systemic reason to adopt such a balanced approach. An ensemble does not require a minimum number of experts to function and can contain as many experts as needed to capture the output-specific logic. All experts impact the ensemble output based on their relative confidence measure.

How much of the input-output correlation is captured in an ensemble is more important than the number of constituent experts. The designer has the freedom (and perhaps the obligation) to focus on designing experts on subsets of the input space that are highly correlated, nonlinear, or otherwise complex. Increasing the number of trained experts in a complex portion of the input space ensures each individual expert remains a domain-specific expert. This enables each expert to remain more simple which aligns with the framework goals of understandability and cooperative decision making. Focusing ensemble growth in this manner has roots in other ensemble design methods including bagging [50] and boosting [51].

The same notion holds true for complex system outputs. An ensemble-of-ensembles may create the top-level output using many lower-level ensembles that each focus on separate portions of the signal space. Per-expert and per-ensemble understandability is critical when autonomous systems cooperate with humans. For many-dimensional spaces that include a mix of quantitative and qualitative inputs and intermediate outputs, per-subsystem simplicity is driven by sharing the problem among many peers.

5.3.3 Parameterization

The structure and design of a fuzzy expert system includes multiple layers of tunable parameters a designer can use to modify the system. At the highest level, changing the signals that an expert can access is one way to drive system specificity. Adding to an expert's signal subspace may increase its ability to reason about complex problems. It is up to the designer to make trade decisions when adding to or limiting the per-expert signal space.

Another layer in the fuzzy system is the fuzzy sets that subdivide each input signal. More fuzzy sets per signal leads to higher granularity and complexity in the possible logic space. Each fuzzy set is a function with the only requirement to map the input signal to $[0, 1]$. These functions are parameterized to activate on parts of a signal domain.

Fuzzy rules are yet another layer within the fuzzy expert system. The rules that make up the expert logic are made from connections between the antecedent fuzzy sets to the consequent fuzzy sets. Which combinations of antecedents map to which outputs is the main method a designer has for capturing input-output correlations. Basing a rule on more antecedents or antecedent groupings may provide additional constraints on a specific logic case that ensures appropriate activation for a given signal space.

Each of these layers (signal subspace, fuzzy sets per signal, activation function parameters for each fuzzy set, and fuzzy rule structure) can be described through careful parameterization. This becomes important in the later stages of design when performance measurement and optimization is considered. A reasonable choice is to hold all but one of these parameter layers constant during the optimization phase to better control and understand how changes manifest in the outputs.

5.3.4 Redundancy and Overlap

In a standard HEADS system, the designer intends to cover as much of the input-output correlation as possible while maintaining low complexity. This manifests through minimizing the size of each layer within the system, including per-expert signal space, rule set, and fuzzy sets, and also the number of experts within each ensemble.

When the system must become more complex to better model the problem, the designer may need to add subsystems (ensembles or experts in an ensemble) that overlap with its peers. This is driven by the complex correlations in the problem space and the approach to modeling through many cooperative but less complex subsystems. It is logical to start by covering as much of the problem without overlap since each new expert or ensemble would solve a previously unhandled state space realization. But, as total coverage of the problem increases so does the difficulty in finding new independent parts of the problem.

Adding an expert that overlaps with another expert in its ensemble can take many forms but typically means starting with the same signal space and adding or replacing a signal (and the associated fuzzy sets and rules). This leaves most of the original expert intact meaning the two cover mostly the same signal space. The resulting new expert provides some redundancy to the original expert which can be useful if certain rules or signal subdivisions are known to show good performance. The major benefit of such slight modification is a completely new set of correlations to explore that incorporate the new signal and fuzzy sets.

5.4 Notes on Agent Theory and Architecture

As discussed in Section 2.4, the proposed framework is a layered architecture with logic-based core functions that incorporates agent theory including attributes of belief, obligation, self-confidence, and selecting a 'good-enough' option quickly.

Belief and obligation are captured through the fuzzy expert rule and fuzzy set design. What a specific fuzzy expert believes to be true depends on how its signal space activates fuzzy sets. A fuzzy set with high activation corresponds to that expert believing the input signal matters. Fuzzy set activation directly impacts rule activation where specific rules will dominate given the signal space realization which corresponds to the expert believing it knows what to do. How the specific rules and fuzzy sets are implemented during the design phase holds the expert system obligations, where the rules can be designed to activation or avoid specific actions under certain circumstances. The expert system is beholden to its peers in the ensemble, and the system as a whole, to act on its beliefs to achieve its obligations. The holistic system incorporates the beliefs of its component parts to take action based on its component obligations.

Agent self-confidence is shown through the expert system self-confidence that propagates through expert output mixing. An expert that has fuzzy sets and rules with high activation over a portion of the signal space will show high confidence when that signal space is active. The ensemble has a similar mechanism when it generates the mixed output and confidence, where the combination of each constituent expert's output based on that expert's self-confidence measure is analogous to a probability distribution with variance. If the component experts' outputs are varied across the output domain and each has similar self-confidence, the mixed output will have low confidence due to the large variance. Conversely, if the expert systems agree on an output with high confidence, then the mixed output will have low variance corresponding to high confidence in the cooperative group's result.

The concept of selecting an action quickly that might not be optimal but is good enough is known as satisficing (opposed to optimizing). Creating a system that satisfices rather than optimizes depends on how uncertainty is handled and what methods are available to reduce uncertainty. Fuzzy expert systems handle non-stochastic uncertainty by computing degree of fuzzy set activation over signal domains. To reduce the uncertainty means to minimize the overlap between fuzzy sets on the signal domain which forces one fuzzy set to activate while neglecting the others. This defeats the purpose of fuzzy sets which are intended to have overlap to capture uncertain state transition boundaries. Therefore it is counterproductive to reduce uncertainty at the fuzzy set level. Fuzzy expert uncertainty is a feature. Due to the overlapping between fuzzy sets and the partial activation of multiple output fuzzy sets an optimal output is not guaranteed. However, redemption comes from the fuzzy expert system computational complexity. Evaluating fuzzy set activation functions, evaluating If-Then logic, combining output fuzzy sets, and finding the bisector of area of the resultant output set are inexpensive calculations. There is no recursive search or optimization to perform to obtain a fuzzy system output. The fuzzy expert system thus exhibits satisficing, where the output is derived from logic based on uncertainty but is achieved at a rapid pace.

Chapter 6: HEADS System Implementation and Analysis

Implementing the hierarchical ensembles of autonomous decision systems framework requires understanding the underlying system. The HEADS design methodology includes a baseline approach to identifying necessary signals and components followed by iterative introspective analysis and modification. Top-level data modeling informs bottom-up design. Analysis is done to measure understandability, complexity, and performance. The general design process follows:

1. identify input, intermediate derived output, and terminal output signals
2. subsystem experts define logic rules for driving subsystem performance according to the available signal space
3. expert systems are fully parameterized mapping signal spaces to rule antecedents and consequents
4. system performance and understandability metrics are assessed
5. iterative updates to each layer are made to achieve design and performance goals

The rest of this chapter outlines this process in detail.

6.1 Define Early, Update Often

Capturing system objectives is at the forefront of the design problem. Assigning measurement signals to operational goals or system objectives is a clear way to start tracking these data for monitoring. How these data are measured is outside the scope of this dissertation but common methods may include raw telemetry data, activity counters, or estimated parameters. Once the performance tracking data model is initialized the designer must account for system inputs and outputs.

When it comes to large complex systems-of-systems it may be logical to approach subsystems as independent decision making problems. Each subsystem subject matter expert is likely better equipped to characterize and present the data model for their problem domain. A power management expert will know the necessary and sufficient signals to track for maintaining battery state of charge and knowing when current draw exceeds limits but may not know how to describe these data for optimal payload operations. The result is modular in that each subsystem derives its expertise from domain specific knowledge. On the other hand, there is value to outside opinions and collective problem solving. Approaching the data modeling problem holistically among subsystem experts may provide further depth of insight and control.

The rest of the system parameterization defines logic and adds detail to the captured data model and input-output relationships. For each expert, the fuzzy rule base provides the logical connection between input and output signals. The fuzzy set subdivisions map rule antecedents to specific portions of the input signal domain.

A first effort result for the system data model and the internal parameterizations should not aim to be an exhaustive list of signals or perfect parameter mappings. Limiting this list to reasonable signals rather than all possible signals helps scope the initial design problem. The full list of signals can always be appended with derived signals. It is also likely that later stages of system development will identify data gaps or excess leading to necessary data model or rule base updates. Each modification updates the accessible data for logic which may result in identifying further intermediate signals to add to the signal space. Iteration can expand or contract these data models and expert parameterizations as needed.

6.2 Outward-In Approach

The design approach discussed so far encompasses an initial top-down data model design to capture system requirements and then a bottom-up rule design to map signals to decision making. This outward-in approach leads to a good mix of understanding system possibilities and constraints and how they impact the decision space. The system structure and per-expert rule bases incorporate these possibilities and constraints.

6.2.1 Defining the Signal Space

The signal space $\mathbf{Z}$ includes data inputs $\mathbf{x}$ and intermediate derived and terminal system outputs $\mathbf{y}$. Overall system requirements and structure define what signals are available.

6.2.1.1 Inputs

Inputs are drawn from data-producing devices or modules throughout the system and may include raw telemetry, mission goal definitions, guidance navigation and control data, and metadata about the network and an agent's neighbors. The input space captures non-decision based data about the agent's state and environment.

6.2.1.2 Intermediate Outputs

The output space combines both intermediate and terminal outputs. Intermediate outputs are mid-layer logical conclusions drawn from the signal space but not directly needed for action selection. These intermediate values are analogous to observable state variables that are used within an estimation filter to yield control feedback signals. The utility of such intermediate signals comes from multi-objective logic, where several indirect factors may be necessary to compare before arriving at a conclusion.

6.2.1.3 Terminal Outputs

Terminal outputs are those that are used for action selection. The terminal output space is multidimensional since there are often many decisions to make concerning different parts of a complex system. There may be many decisions for a specific action, for example operating a sensor may require warmup time, calibration, on-off triggering, focusing actions, and other control signals that may depend on the current state or system objectives.

6.2.1.4 Signal Space Modularity

The signal space for decision making in the HEADS framework includes all inputs and outputs. The reason behind this is aligned with the idea of blackboard systems where many consumers might depend on any number of signals throughout the system. This modular approach leads to each expert system within an ensemble being capable of using any subset of inputs and outputs for logic. The general availability of the data makes it easy to extend the system through the addition of experts to ensembles or ensembles to the system to provide new outputs. This attribute leads to modularity without the need to retrain the entire system.

6.2.1.5 Recommendations for Identifying Signals

In the context of autonomous decision making, every signal available throughout the system may not be relevant or important. Scrutinizing questions can help designers identify the necessary and sufficient signals during early stages of design. Terminal outputs drive system actions. Intermediate outputs can help clarify the internal system logic. Examples of leading questions and answers are shown in Table 6.1.

Table 6.1 Example questions and answers for identifying the signal space

Question	Possible Answers
What are the operating states of the device?	on, off, standby, warmup, sensing, communicating, processing
What would prevent safe operation of the device?	low power, proximity to another system, exceeding duty cycle
What factors might drive a decision to operate the device now or later?	duration until next known observation target, utility of gathering data of non-specified targets, the amount of power reserves available, whether a neighboring system is operating its device
What sequence of actions is favorable for achieving mission goals?	taking observations and sharing post-processed data, sharing telemetry archives before purging memory, recalibrating between observation of high-value targets, waiting for operator confirmation before proceeding with the next action
When is it worth risking activity at or near safety boundaries?	when anticipated reward exceeds some threshold, when no other nearby system can achieve the task in a reasonable amount of time

6.2.2 Defining Goal Based Operation

Once the available signals are known through signal space identification, the designer must map the inputs to outputs through logic rules. These rules must account for system objectives and are informed by system performance metrics.

The designer defines rule antecedents joined by logical operators that map subsets of the signal space to an output consequent. As many rule antecedents as necessary are used to describe the logic rules for a particular expert. However, it is recommended to maintain coherence in the rule set and align the logic with a particular view of the problem. Doing this adheres to the design mentality of collaborative decision making by cooperation between many domain experts. It may be beneficial to ask separate designers to create expert systems to solve the same problem. Each designer brings a different perspective and the resulting expert system will add to the robustness of the ensemble.

When selecting signals to use as rule antecedents, some require more pre-processing than others. Quantitative signals require translation into linguistic variables. For example, a temperature signal might be used in a logic rule via the 'low temperature' qualifier. Qualitative signals may be used directly since they tend to describe a state using linguistic variables already. The qualitative signal 'operator is uncomfortably warm' can be used as a logic antecedent without modification. Each logic antecedent is mapped to an input fuzzy set $\mu_{j,p,u}$ via the linguistic variable.

The HEADS framework assigns a single output signal to each ensemble. All ensemble experts try to answer the same question of whether to activate the output signal. Each system output may include qualifiers to its activation. For example, 'soon' or 'now' are qualifiers to the decision to turn on a sensor. Rather than treat each of these qualified outputs as individual outputs, they are handled through subdivision of the outputs into activation domains. These qualified outputs are each a logic consequent to be considered by the expert systems in an ensemble and are mapped to an output fuzzy set $\nu_{j,p,v}$.

Linguistic variables representing the selected antecedents and consequents are used for rule making and are joined by standard logic operators including AND, OR, and NOT:

- AND: (A and B) $\equiv \min(A, B)$
- OR: (A or B) $\equiv \max(A, B)$
- NOT: (not A) $\equiv (1 - A)$

Logic statements can be grouped parenthetically to enforce an order of operations:

- $(A = 0.5$ or $B = 0.8)$ and $C = 0.2 \rightarrow 0.2$
- $A = 0.5$ or $(B = 0.8$ and $C = 0.2) \rightarrow 0.5$

Each rule then takes the standardized format ‘IF antecedents THEN consequent’ which is designed to capture human expertise quickly.

Multiple consequents can be considered by a given expert system. Output fuzzy set combination and output value computation done through aggregation and defuzzification routines.

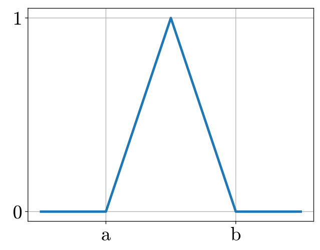
6.2.3 Defining Rule Activation Boundaries

Once linguistic variables are defined for the rule antecedents and consequents they are collected by the input signal used. Each linguistic variable corresponds to a subset of the input or output signal domain. The linguistic variable activates according to a function defined on its subset of the signal domain. Common functions including triangles, trapezoids, and sigmoids are shown in Figure 6.1.

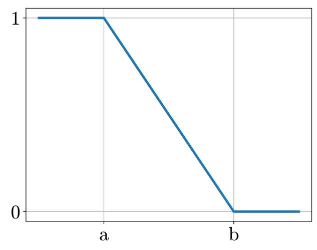
The fuzzification and defuzzification processes define how the fuzzy system will process information and deliver an output and confidence value.

Fuzzification is the mapping of input signals through activation functions:

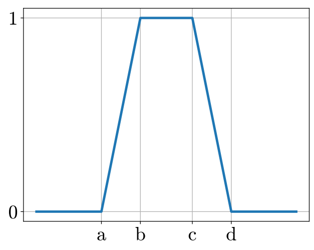
$$\mu_{j,p,u}\left(z_{j,p,k} \in \mathbf{z}_{j,p}\right) \rightarrow [0, 1] \tag{6.1}$$



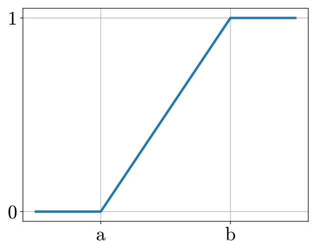
(a) ‘TRI’



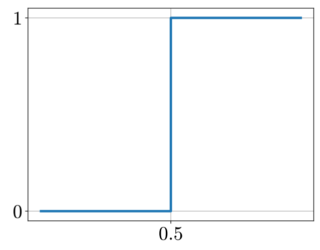
(b) ‘LTRAP’



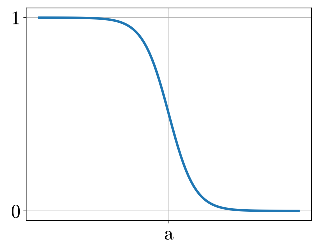
(c) ‘TRAP’



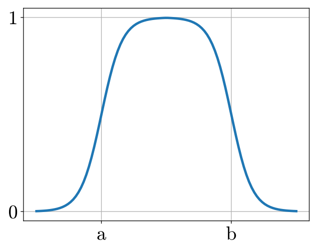
(d) ‘RTRAP’



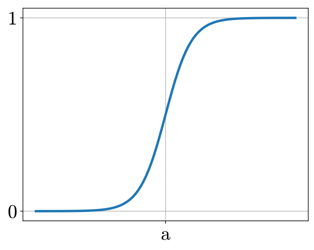
(e) ‘BINARY’



(f) ‘LSIG’



(g) ‘SIG’



(h) ‘RSIG’

Figure 6.1 Fuzzy set activation functions

A fuzzy set activation function impacts rule activation through definition of state transition behavior. A steep edge corresponds to a narrow state transition window and effectively indicates more certainty in the boundary. It is through these fuzzy set activation function parameters that fine-tuning system behavior is achieved. For minor changes, a manual process may be appropriate, since the designer can identify fuzzy sets and rules that are misfiring. Through direct understanding of what each subject variable means the designer can adapt parameters to achieve desired performance.

Fuzzy sets activate when their input signal lies within the domain of the fuzzy set activation function. The impact on rule consequent activation is determined by the fuzzy logic operators AND, OR, and NOT. The combined antecedent activation $r_{j,p,w,ANT}$ acts as a range limit for the consequent fuzzy set activation $\nu_{j,p,v}$:

$$r_{j,p,w,CON} = \min_{y \in y_j} \left[\nu_{j,p,w}(y), r_{j,p,w,ANT} \right] \tag{6.2}$$

Careful definition of the fuzzy set parameters is required to achieve the desired rule activation for a given system and environmental state.

6.3 Introspection: Signal Space Coverage

HEADS includes many experts across multiple levels of ensembles within the system. The input space coverage metrics are useful for understanding how specific or widespread a given expert's or ensemble's input utilization is. Binary coverage metrics can grant insight into whether an ensemble covers an appropriate amount of the system's signal space. A weighted metric allows relative signal coverage to be assessed. Signals with large weighted coverage statistics might indicate to a system designer that it is worth investing to increase the accuracy and precision of that signal or to subdivide the signal to better capture its components. Signals with sparse coverage may indicate unimportant or unnecessary data that could be removed from the system with little impact on performance.

The expert binary coverage vector for expert syste p in ensemble j is defined:

$$\alpha_{j,p} = [\beta_1, \beta_2, ..., \beta_{m+n}]^T \tag{6.3}$$

where

$$\beta_k = 1 \text{ if } z_k \text{ used by } s_{j,p} \tag{6.4}$$

with index k indicating position in the system's signal space $\mathbf{Z}$, and the result a vector of ones and zeros indicating signal use within expert $s_{j,p}$.

A weighted coverage vector can be made by summing occurrences of an input use by individual rules within an expert:

$$\overline{\alpha}_{j,p} = [\sigma_1, \sigma_2, ..., \sigma_{m+n}]^T \tag{6.5}$$

with

$$\sigma_k = \sum_{w=1,2,...,w_{j,p}} \begin{cases} 1 & \text{if } z_k \text{ used by } r_{j,p,w}\left(\mathbf{m}_{j,p}\right) \\ 0 & \text{otherwise} \end{cases} \tag{6.6}$$

where a rule is a function of several fuzzy sets that each depend on a signal that may be a subset of the expert's signal subspace:

$$r_{j,p,w}\left(\mathbf{m}_{j,p}\right) \equiv r_{j,p,w}\left(\mathbf{z}_{j,p,w} \subseteq \mathbf{z}_{j,p}\right) \tag{6.7}$$

Both binary and weighted coverage can be applied to the ensemble. The ensemble binary coverage vector for ensemble j is made from combining the per-expert binary coverage vectors:

$$\alpha_j = \left[\beta_1, \beta_2, ..., \beta_{m+n}\right]^T \tag{6.8}$$

where

$$\beta_k = \max_{p=1,2,...,p_j} \alpha_{j,p,k} \tag{6.9}$$

The ensemble weighted coverage vector similarly sums the weights from the per-expert weighted coverage vectors:

$$\overline{\alpha}_j = \left[\overline{\sigma}_1, \overline{\sigma}_2, ..., \overline{\sigma}_{m+n}\right]^T \tag{6.10}$$

where

$$\overline{\sigma}_k = \sum_{p=1,2,...,p_j} \overline{\alpha}_{j,p,k} \tag{6.11}$$

The system-level coverage metrics are another straightforward extension:

$$\alpha = \left[\beta_1, \beta_2, ..., \beta_{m+n}\right]^T \tag{6.12}$$

where

$$\beta_k = \max_{j=1,2,...,m} \alpha_{j,k} \tag{6.13}$$

and

$$\overline{\alpha} = \left[\overline{\sigma}_1, \overline{\sigma}_2, ..., \overline{\sigma}_{m+n}\right]^T \tag{6.14}$$

where

$$\overline{\sigma}_k = \sum_{j=1,2,\ldots,m} \overline{\alpha}_{j,k} \tag{6.15}$$

6.4 Introspection: System Complexity Metrics

System complexity arises from nonlinearity, uncertainty, degrees of freedom, and non-separability. These characteristics are inherent to the class of problems considered here. By definition the network of distributed systems-of-systems (NDSS) has many degrees of freedom and contains a large number of interrelated processes that exhibit nonlinear and difficult to model boundary behaviors. The main approach within HEADS to handle this complexity is through multi-agent cooperative decision making. Each individual agent, or expert system, contains its own model of a portion of the overall problem space. Combining these sub-parts that are by design less complex than the whole allows the designed system to capture and address most of the underlying system's complexity.

When designing for understandability, reducing complexity helps ensure systems require minimum analysis and translation. Thus, it is useful to define complexity metrics that can identify expert systems or ensembles that may need to be reconfigured into more subcomponents. The overall balance and range of complexity scores will differ based on the problem at hand. However, the designer can use these metrics to compare iterations of the same system used to solve a particular problem.

Expert system metrics focus on the variation in information used or created by the expert. The first metric is the number of rules an expert system contains:

$$\psi_{j,p,rules} = w_{j,p} \tag{6.16}$$

where j indicates the ensemble, p the specific expert in the ensemble, and w the rule index for an expert. The number of fuzzy sets in use by an expert complements the rule complexity metric:

$$\psi_{j,p,fsets} = u_{j,p} \tag{6.17}$$

where u is the fuzzy set index for an expert. The number of fuzzy sets defined for a given input signal is

$$\psi_{j,p,z_l} = \sum k = 1, 2, ..., u_{j,p} \begin{cases} 1 & \text{if } \mu_{j,p,k} \text{ uses } z_l \\ 0 & \text{otherwise} \end{cases} \tag{6.18}$$

where z_l is one signal in an expert's signal vector $\mathbf{z}_{j,p}$ and μ is the fuzzy set activation function defined on the signal domain. The complexity due to the number of antecedents used by a rule, sometimes referred to as rule compactness, is

$$\psi_{j,p,w,ANT} = \sum_{k=1,2,...,u_{j,p}} \begin{cases} 1 & \text{if } \mu_{j,p,k} \text{ used by } r_{j,p,w} \\ 0 & \text{otherwise} \end{cases} \tag{6.19}$$

which, when considered at the expert system level, aggregates over all rules used by an expert either through the maximum or average:

$$\psi_{j,p,ANT} = \max_{k=1,2,...,w_{j,p}} \psi_{j,p,k,ANT} \tag{6.20}$$

$$\psi_{j,p,\overline{ANT}} = \frac{1}{w_{j,p}} \left[\sum_{k=1,2,...,w_{j,p}} \psi_{j,p,k,ANT} \right] \tag{6.21}$$

A metric capturing rule activation is useful as measure of a rule's impact on an expert system's reasoning. Rule activation is the result of fuzzy evaluation of the If-Then logic rules resulting in a rule consequent:

$$\psi_{j,p,w,ACT} = r_{j,p,w,CON} \tag{6.22}$$

where $r_{j,p,w,CON}$ is described in Equation 6.2. This metric is most useful when considered over time, since rule activation changes as system and environment states change. A rule similarity metric to measure how different or the same a rule is compared to another rule with a different consequent results in a rule similarity matrix:

$$\mathbf{\Psi}_{j,p} = \begin{bmatrix} 1 & \lambda_{1,2} & \lambda_{1,3} & \cdots & \lambda_{1,w_{j,p}} \\ \lambda_{2,1} & 1 & \lambda_{2,3} & \cdots & \lambda_{2,w_{j,p}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \lambda_{w_{j,p},1} & \lambda_{w_{j,p},2} & \lambda_{w_{j,p},3} & \cdots & 1 \end{bmatrix} \tag{6.23}$$

where

$$\lambda_{k,l} = \frac{1}{w_{j,p}} \left[\psi_{j,p,k,\overrightarrow{ANT}} \cdot \psi_{j,p,l,\overrightarrow{ANT}} \right] \tag{6.24}$$

$$\psi_{j,p,w,\overrightarrow{ANT}} = \left[\beta_1, \beta_2, ..., \beta_{u_{j,p}} \right] \tag{6.25}$$

$$\beta_k = \begin{cases} 1 & \text{if } \mu_{j,p,u} \text{ or } \nu_{j,p,v} \text{ used by } r_{j,p,w} \\ 0 & \text{otherwise} \end{cases} \tag{6.26}$$

The above metrics can be summed over all experts within an ensemble to generate ensemble-level metrics:

$$\psi_{j,rules} = \sum_{k=1,2,...,p_j} \psi_{j,k,rules} \tag{6.27}$$

$$\psi_{j,fsets} = \sum_{k=1,2,...,p_j} \psi_{j,k,fsets} \tag{6.28}$$

$$\psi_{j,z_l} = \sum_{k=1,2,...,p_j} \psi_{j,k,z_l} \tag{6.29}$$

where z_l is one signal in the ensemble's signal vector $\mathbf{z}_j$. The rule compactness metric is most meaningful as either the maximum or average across all experts within the ensemble:

$$\psi_{j,ANT} = \max_{k=1,2,...,p_j} \psi_{j,k,ANT} \tag{6.30}$$

$$\psi_{j,\overline{ANT}} = \frac{1}{p_j} \left[\sum_{k=1,2,...,p_j} \psi_{j,k,\overline{ANT}} \right] \tag{6.31}$$

Note that the average is computed from the already-aggregated per-expert average rule compactness. This approach was selected instead of recomputing the average of all rule compactness between ensembles to ensure each ensemble is given equal weighting for the combined metric. Recomputing the average using the rules is a valid approach that would yield slightly different results. Finally, the ensemble-level rule activation has no meaning other than as already captured by the ensemble's mixed output y_j as built from individual expert outputs $y_{j,p}$.

HEADS system-level complexity further aggregates the ensemble metrics:

$$\psi_{rules} = \sum_{k=1,2,\ldots,m} \psi_{k,rules} \tag{6.32}$$

$$\psi_{fsets} = \sum_{k=1,2,\ldots,m} \psi_{k,fsets} \tag{6.33}$$

$$\psi_{z_l} = \sum_{k=1,2,\ldots,m} \psi_{k,z_l} \tag{6.34}$$

where z_l is one signal in the system signal set $\mathbf{Z}$. Once again rule compactness is meaningful as a maximum or average:

$$\psi_{ANT} = \max_{k=1,2,\ldots,m} \psi_{k,ANT} \tag{6.35}$$

$$\psi_{\overline{ANT}} = \frac{1}{m}\left[\sum_{k=1,2,\ldots,m} \psi_{k,\overline{ANT}}\right] \tag{6.36}$$

6.5 Introspection: Sensitivity Analysis

The variation in expert system and ensemble outputs as a function of variations or uncertainty in the signal space is an important metric for understanding the nonlinearities within the system. However, sensitivity analysis for a multi-dimensional multi-agent decision making system is not straightforward. Difficulties arise due to the interdependency between signals through rule evaluation and activation. Rule antecedent dimensionality also poses challenges to displaying sensitivity data where any rule with more than two antecedents cannot be holistically displayed.

The most common solution to the interdependency and dimensionality problems is to hold all signals at their nominal values and vary one at a time to measure its impact on the outputs. This is the chosen approach to performing sensitivity analysis for the HEADS framework. Where a single nominal state is not meaningful for the system the analysis can be repeated with each combination of states. An example of why this can matter is where the output action is activating a device. Specific logic may be built into the system to perform certain maintenance actions when the device is active. Deactivating the device may enable a different internal set of rules. How the system behaves as a function of deviations and uncertainty thus depends on two fundamental system states and the designer may wish to treat each state's sensitivity analysis separately.

A primary benefit of sensitivity analysis within the HEADS framework is insight into signals with great impact over system behavior. Identifying such signals can inform the system designer to potential sources of instability during operation leading design of rules to avoid those states. When the sensitivity is attributed to uncertainty in a signal, better data can be sought or safeguards built in through specific rules to identify and mitigate negative impacts when the system is subject to such uncertainty.

Chapter 7: HEADS Performance and Optimization

Computational performance and decision making performance are two major perspectives that are considered complementary to overall HEADS system performance. Computational performance must be considered for application to the network of distributed system-of-systems problem due to limited compute power or implementation architecture constraints. Decision making performance is more aligned with whether the system behaves as expected. The rest of this chapter concerns these two perspectives on computational and decision making performance and concludes with optimization techniques.

7.1 Computational Performance

At its core, computational performance concerns the number and speed of the necessary computations and the memory required to hold the necessary data. Profiling the software implementation of the HEADS framework is beyond the scope of this dissertation. However, as HEADS is designed with computational simplicity in mind the resource and computational requirements are worth discussing.

7.1.1 Resource Requirements

The HEADS framework relies on memory for two main purposes including storing static and dynamic information and intermediate memory for performing reasoning tasks. The static information consists of the expert system definitions and system connectedness design. Dynamic resources are used to hold signal variables and intermediate data products during algorithm execution including rule aggregation, output defuzzification, and ensemble-level output mixing.

The rules and fuzzy sets are static for the HEADS framework that uses fuzzy systems as logic experts. Fuzzy sets are defined as parameterized functions and require static memory to hold the function type and its parameters. Rules are mappings between fuzzy set activation functions and are reduced to minimum and maximum comparisons between fuzzy set function evaluations. There is no formal stored representation of the hierarchical structure. The structure is instead contained within the fuzzy set definitions defined for specific input or intermediate output signals.

The signal space is the largest collection of variable memory used by the HEADS framework. Each input signal holds the variable value and each output signal holds the variable value and a confidence measure. Other relevant metadata is available including time of last signal modification. Algorithm execution also requires variable data to support:

- fuzzy system rule evaluation using variable data from signals processed through the fuzzy set activation functions,
- fuzzy system rule output aggregation via minimum and maximum comparisons,
- fuzzy system output defuzzification and confidence value calculation via computation of the bisector of area under the aggregated output fuzzy set, and
- ensemble mixing of the expert system outputs.

The variable data used for algorithm execution drives peak memory use above the static baseline used for system definition.

7.1.2 Computational Requirements

The computational complexity of the HEADS framework using fuzzy systems is generally low. Most of the calculations are recursive minimum and maximum comparisons between variables. The fuzzy set activation functions can be as simple or complex as required to meet performance, although most consist of only a handful of multiplication and division operations. The defuzzification algorithm is often the most computationally complex since common methods require integrating under the output fuzzy set. Numerical integration can be achieved through simple multiplications and additions but the total number of operations can be large in support of sufficient output accuracy.

The most impactful design feature of the HEADS framework is the reduction in per-expert system size compared to a monolithic fuzzy system of similar signal space dimensions. As discussed in Chapter 4, marginal reductions in the per-expert input space dimensionality quickly result in total complexity reduction of 90% or more. Of course, the monolithic and reduced systems are unlikely to require or implement fully-connected input-output combinations so this assessment is more intellectual than practical. Still, efficient computation at the system level is supported by minimizing each individual expert system's requirements.

Another computational efficiency to consider is different data rate requirements for specific decisions to be made. The decision to activate a sensor device that requires significant warm-up time does not need to be re-made continuously during said warm-up time.

7.2 Decision Making Performance

The expert systems within an ensemble each contribute to selecting actions to meet operational goals. The HEADS system seeks to achieve mission objectives as written into the rule set. When multiple objectives are contradictory the decision framework considers the holistic signal space and selects the action to prioritize. Deconfliction can also be explicitly designed into the rules.

Performance objectives for a network of distributed systems-of-systems may include:

- maximizing raw time spent sensing, amount of data collected, or amount of data delivered,
- completing tasks aligned with a single system's goals while not interfering with or impeding other systems,
- maintaining a minimum state of readiness to be able to act quickly on new tasks,
- quickly identifying and proactively mitigating fault conditions,
- rebalancing activity to cover for known unrecoverable faults on other systems, and
- supporting network-wide objectives and performance metrics.

Each of these objectives can be assigned a numerical score or cost function. The entire system can be simulated and performance measured for design and evaluation purposes.

The decision of how to apply scoring and what metrics to score are critical for design iteration and optimization. Through simulation and evaluation, these scoring metrics inform the designer about decision making performance. Approaches to tuning performance include both manual and automated editing. Manual editing enables more insight into how the system changes and can more easily account for logical consistency during rule or parameter modification but can be slow and expensive. Automated tuning relies entirely on the numerical score to determine fitness of a specific system parameterization. If a particular action or outcome is more important to achieve, it should be given a higher weight during scoring. If actions or outcomes are to be avoided, assigning a zero or negative score is appropriate. Whichever method is used, the scoring metrics provide the baseline for editing.

7.3 Optimization

Optimization is considered from the perspectives of decision making and simulated performance. Several competing factors drive specific optimization implementation including maintaining the original design, general understandability, and driving the system toward maximum performance. Evolutionary methods are used for their strengths concerning gradient-free optimization and ability to cover a wide search space in concert with fine adjustments near local maxima.

7.3.1 Optimizing for Human Factors

The HEADS framework supports understandability through capture of human expertise in linguistic variables and logic rules. Comprehension starts with reasoning through linguistic variables as abstractions to numerical data. Where and how the fuzzy set activation functions are placed on the signal domain is first done by a human designer and captures domain-specific relations between the resulting linguistic variables. For example, the variable 'low temperature' should cover a lower signal domain subset than 'high temperature.' Allowing an optimization technique to rearrange these fuzzy sets would lead to illogical orderings that reduce understandability. This is one example of a constraint on optimization that supports comprehension. Other possible constraints include:

- setting an upper bound for fuzzy set overlap,
- ensuring fuzzy set domains cover realistic signal domains,
- reducing rule opposites,
- ensuring sufficient differentiation between rules with different consequents, and
- balancing expert system rule and fuzzy set additions that increase the complexity metrics outlined in Chapter 6.

7.3.2 Optimizing for Performance

Simulation performance based on numerical scoring or cost functions forms the basis for automated fuzzy set parameter optimization. Iterative system parameter updates and simulation runs lead to identifying parameter sets that increase overall performance. Then, the top-scoring parameterization can be implemented directly on the target system. A strictly numerical approach to optimization can be done without the constraints mentioned in the previous section. Of course, this may lead to reduced human comprehension of the underlying system behaviors. This is a design trade available to the system developer.

7.3.3 Evolutionary Optimization

Gradient-free optimization techniques include random search, simulated annealing, and evolutionary algorithms. These techniques are powerful when objective functions have many local maxima and necessary when computing a gradient for the objective function is difficult or impossible. For the HEADS system the objective function is a numerical score or cost function that rewards actions and positive system states. The signals used as inputs to the scoring function are not directly linked to the parameters that are optimized, making gradient-free techniques necessary for automated optimization. The optimization method of choice for tuning HEADS systems is a variation on genetic algorithms (GA) using real-valued chromosomes. Once an initial HEADS system is designed and a simulation or other test methodology prepared, the GA can be used to iteratively generate new system parameterizations, or phenotypes, for testing.

7.3.3.1 HEADS as a Genetic Algorithm Problem

Real-valued parameters are used for the HEADS genetic optimization approach. The fuzzy set activation function parameters form the genotype to be modified by the genetic algorithm optimization routine. A genotype describes all possible tunable parameters. A phenotype describes a specific individual parameter set within the population of candidate systems. Each new phenotype represents an entire system parameterization to be tested through simulation.

Each activation function parameter set contains a small number of values, typically two (2) to four (4). The genotype, or set of properties to adapt, is created from real-valued parameters. This enhances direct comprehension of the internal process during optimization. The phenotype, or specific instance of the genotype representing an entire system, remains relatively compact even for systems with several experts in each of many ensembles.

7.3.3.2 Genetic HEADS Primer

A designer develops the HEADS system capturing necessary and sufficient signals, creating rules mapping inputs to outputs, and designing the rule activation boundaries by defining fuzzy set activation function parameters. Selection of performance metrics and creating a scoring methodology then support simulation-based performance analysis. Baseline performance is increased through design introspection and manual system modification. Further simulation of the modified system may lead the designer to determine that fine-tuning could result in better performance. The genetic algorithm tuning process is begun.

First, the designer exports the system's initial parameter set as the first population member. This can be done in many ways but the most straightforward is to create a dictionary containing the general system hierarchy (ensembles, experts, fuzzy sets, fuzzy set activation function parameters):

1. y_1
   (a) $s_{1,1}$
      i. $battery\_low = [a, b]$
      ii. $battery\_nominal = [a, b, c, d]$
      iii. $battery\_high = [a, b]$
      iv. ...
   (b) $s_{1,2}$
      i. ...
   (c) ...
2. y_2
   (a) ...
3. ...

This dictionary structure is the genotype. When the genotype contains a set of values describing an individual system parameterization it is a phenotype.

The first phenotype is used to seed an initial population. Each parameter in the initial phenotype is used as the mean for a stochastic expansion. This results in each population member having a different phenotype that is similar to the initial member. If optimizing under constraints for understandability, these constraints would be applied during parameter generation, and new parameters would be modified according to the constraints to ensure each phenotype is valid.

Once the initial population is created, each phenotype is evaluated according to the same simulation to ensure valid scoring comparisons can be made. Each phenotype is given its score and the top performers across all phenotypes are identified. These top scoring phenotypes are used during the evolutionary phase to create new population members through stochastic recombination and mutation. The newly created phenotypes are simulated and scored. The process is repeated until either a minimum score threshold is met or a maximum number of iterations is reached.

The first evolutionary method creates new phenotypes through stochastic mixing of two members of the top-scoring group. A binary vector of length equal to the total number of fuzzy sets is randomly created. The child phenotype is generated from parameter sets provided by either parent depending on the binary vector value. Each fuzzy parameter set is copied in its entirety from a single parent. This ensures the child phenotype is valid and adheres to the constraints applied during evolution without further modification. The result is one new phenotype for every pair subject to recombination.

The next evolutionary method takes a single member from the top performing group and copies its entire phenotype into a new individual. This new individual then undergoes mutation where a relatively small number of individual parameters throughout its phenotype are stochastically modified. This has the potential to create an invalid phenotype and so the constraints applied after mutation. The result is a phenotype that is similar to the original but with a few parameters that differ to a possibly large extent.

Tuning parameters to drive genetic diversity include the stochastic variances used to create the original population or mutate parameters. These tuning parameters are typically set so mutation can result in vastly different values to drive population diversity. This encourages search beyond the local maximum near which most population top-performers are likely to be found. The process is repeated a sufficient number of times to either meet some minimum score threshold or maximum iteration count. The GA cannot guarantee the global maximum will be found, although after sufficient iteration a solution that approaches the global maximum is likely.

Chapter 8: HEADS Applied to the NDSS Problem

The hierarchical ensembles of autonomous decision systems framework has thus far been thoroughly examined through analysis and theory related to complexity, autonomous agents, and the need for understandability. This chapter is dedicated to showing the HEADS framework applied to a representative problem from the networks of distributed systems-of-systems (NDSS) class of problems.

The NDSS problem exhibits complexity at both the individual agent level and across the network. Objectives may not be shared or even realizable by all members of the network. Individual agents may have multiple objectives that partially overlap with neighboring agents' objectives. An agent and its neighbor may be fundamentally different in terms of capability or resource capacity. Network dynamics including physical reconfiguration and delays and disruptions to communication further stress cooperating clusters of agents. The size of the networks in question are perhaps the largest source of complexity as optimizing or satisficing for all the other problems becomes more difficult at scale. These and other factors lead to the conclusion that highly-capable per-agent system autonomy is a necessity for current and future networks of distributed systems-of-systems.

The example problem that follows was chosen as the representative member of the NDSS class of problems due in part to its relevancy to current commercial, academic, and government activities in the space domain. The NDSS is made up of a large number of cooperative spacecraft in low Earth orbit (LEO). Each spacecraft has one of several possible payload packages made up of sensors, actuators, communicators, and processors that enable it to be a high performer in its area of expertise. Inter-spacecraft links that provide robust low-bandwidth and sporadic high-bandwidth connectivity enable communication with neighbors. Tasking is done by request and prioritization where specific requests are added to the pool of tasks and each autonomous agent executes tasks according to its on-board decision making. The spacecraft are distributed across multiple orbital inclinations and can all communicate with each other when in range. However, they are also assumed to be from many tranches placed in orbit over time and therefore subject to occasional partial or total failure. The problem becomes efficient tasking and oversight of NDSS performance to maximize its utility and responsiveness. The HEADS framework is implemented as an autonomous agent 'Flight Director' to solve the problem.

The rest of this chapter covers the design of the simulation developed as a test platform for this problem, simplifying assumptions and constraints, and the associated performance metrics used to evaluate solutions. A HEADS system is designed according to the methodology presented in Chapter 6. Introspective analysis is presented including coverage, complexity, and sensitivity measures. The HEADS system is simulated as the on-board autonomous agent for all spacecraft of a particular capability and performance results provided. The as-designed HEADS system is optimized using an evolutionary algorithm as described in Chapter 7. Post-optimization performance is presented and compared to the original system performance. Generalized performance is proven through application of the optimized HEADS system to a simulated scenario not used for training.

8.1 Simulation Design Overview

The specific problem approached through this simulation is that of remote observation of a ground point somewhere on Earth and the subsequent transmission of sensor data to end users. This remote sensing subproblem requires only a subset of the entire orbital network of capabilities. Other systems within the network operate in supporting roles to process and transfer data.

The designer of an operational system tasked with solving this NDSS problem may wish to explore every aspect in great detail. This representative problem contains several interesting and complex subproblems including orbital dynamics, sensor performance across orbital inclinations and altitudes, and communications network dynamics. Each topic has been the subject of papers, theses, and dissertations that provide the background for the simple application of such theory as used here.

Vehicle dynamics are limited to translational point-mass dynamics subject to two-body gravity forces. Atmospheric, solar radiation pressure, and other perturbances are ignored. The gravity model is calculated using the second and third zonal harmonics from [123] and the Legendre polynomials from [124]. Attitude dynamics are ignored.

Each spacecraft type has finite energy storage that is 100% efficient enabling power to be stored and retrieved without loss. Power generation operates at peak efficiency as soon as the solar panels are in line of sight with the Sun. As long as the system has positive stored energy it can continue to operate as driven by the Flight Director autonomous agent.

Interactions with other nodes in the network occur subject to line-of-sight and range constraints. The 'nearest neighbors' include all other spacecraft within 250 km and within line-of-sight. Proximity to a 'communications node' is sufficient to enable delivering sensor data to end users. Proximity to a 'processing node' enables transfer of data for edge processing and combination with other data sources to yield higher-value data.

The simulation itself operates in three main phases as follows:

1. Initialization:

    (a) space vehicle translational dynamics are simulated for the entire simulation duration,

    (b) network connectivity is computed at each step according to the range and line-of-sight constraints,

    (c) the translational and communications network dynamics are used to create per-satellite input vectors for states independent from on-board decision making, for example line of sight to the sun or proximity to a processing node,

    (d) each spacecraft is initialized with a state vector using the pre-computed values where possible and stochastically generated values otherwise, and

    (e) simulated faults are stochastically generated and replace portions of the computed state vectors.

2. Simulated Reasoning:

    (a) all Flight Director systems perform reasoning using their own state vectors,

    (b) action selections are disseminated across the communication network to nearest-neighbors,

    (c) dynamic state variables are computed using the selected actions, any resulting impacts, and the simulated faults, and

    (d) steps 2(a) to 2(c) are repeated until the entire duration is complete.

3. Scoring:

    (a) each spacecraft has its state vector and decision vector analyzed and scored, and

    (b) a total network-wide score is assigned.

8.1.1 Orbital Remote Sensors

The focus on remote sensor operation is interesting because other network support activities tend to be reactionary to data creation. The processing and transfer of data to end users requires data to be created first. This example considers a large network of distributed sensors that must operate with autonomy. Network-level utility is therefore sensitive to autonomous agent performance. Basic network support activities are not the subject of this simulation and are therefore assumed to be more available than the data generation operations.

Two distinct models of space vehicle are simulated for this problem. The first contains a wide-angle optical camera, a narrow-angle optical telescope, and a medium-angle near-infrared sensor. This vehicle type generates solar power at a rate of 21 Watts. The second hosts a wide-angle optical camera, a synthetic aperture radar, and a laser altimeter and generates 42 Watts of power in the sun.

Each individual sensor is modeled as requiring energy to provide valuable data. When the sensor is on it draws constant power, and when it is off it uses energy at a much lower idle rate. There is a standard reward assigned to data from each sensor. Some sensors generate data with increased rewards for continuous use of the sensor.

8.1.2 Network Communications

Translational dynamics are the primary driver behind the space NDSS communication network dynamics. An automated link is assumed to exist whenever two space vehicles are within range and are not occluded. When two sensing spacecraft are neighbors, their inter-satellite communications are robust and zero-latency for transmitting state information. Having a communications node as a neighbor enables high-throughput data transfer into the communications network. The communications nodes are assumed to have sufficient storage and data throughput such that they can always accept data from neighbors. Finally, a processing node neighbor leverages the same data transfer link as used for the communications node but results in a value multiplier for any volume of data transferred to the processor. The processing layer is assumed to transfer its own data product across the network through the communications layer.

8.1.3 Goal Based Tasking

Tasking is modeled as a simple assignment of ground points of interest and a priority or score modifier. The network as a whole is left to ingest, disseminate, and process these tasks. This approach relies on the network to adapt to dynamic task plans without specific instruction on how to do so. The tasks are uplinked from a ground location and spread between spacecraft as the dynamic communications network allows.

8.1.4 Simulated Failures

Partial and total failures are simulated at either the payload or the spacecraft level. Possible failures include excessive power draw when devices are active, transceiver failures that prevent transmitting or receiving, and payload failures preventing the creation of usable data. These failures are applied stochastically to payloads or entire spacecraft with the dynamic effects impacting state and decision vector updates.

These faults are simulated to show how each individual unit is robust to operating in a way that causes total system failure as measured by reducing stored energy to zero. Reactions to these events also show the utility of a distributed network of sensors through the resilience to loss of nodes.

8.1.5 The NDSS: Spacecraft and Orbits

The distributed spacecraft system includes sensing, processing, and communication nodes. While the remote sensing satellites are the focus of autonomous decision making for this study, the other nodes are included to simulate realistic communications network dynamics. The communications and processing nodes are presumed to work automatically and at all times. The impact on sensor nodes is through known instances of physical proximity, i.e., when within range of cross-linking communications. The defining characteristics of the network are number of satellites per orbit, number of orbits, orbit altitudes, orbit inclinations, and the right ascension of the ascending node (RAAN) for each satellite in an orbit.

To illustrate the number of satellites required for a high degree of network connectivity, consider a perfectly circular orbit of altitude a_o around Earth with radius R_e and maximum inter-satellite range r_{max}. The number of satellites needed to meet the range requirements is:

$$N \geq \pi \left[\arcsin \left(\frac{r_{max}}{2\left(R_e + a_o\right)} \right) \right]^{-1} \tag{8.1}$$

For example, at 500 km altitude with maximum communications range of 500 km, the orbit would need at least 87 satellites to ensure contiguity of links. This assumes a perfectly even distribution of spacecraft throughout the orbit. To achieve network connectivity and a reasonable distribution of sensing and processing nodes, many spacecraft in multiple orbits are required. Table 8.1 shows the distribution of the 540 spacecraft that are simulated for this example. The ‘spread’ metric indicates whether the spacecraft start evenly spread or stochastically placed around the orbit. Figure 8.1 illustrates the distribution of the entire constellation in an Earth-centered inertial frame. Figures 8.2-8.4 show each component spacecraft distribution throughout the simulated orbits for communication (COM), processor (PROC), and sensor (SEN) nodes respectively. The orbits were selected at common altitudes and inclinations typical for low Earth orbit satellite deployment. The RAAN parameter was assigned to provide moderate separation between orbital planes to simulate a well-distributed network.

Table 8.1 Simulated spacecraft and orbit definitions

Orbit Index	Spacecraft Function	Number of Spacecraft	Altitude (km)	Inclination (degrees)	RAAN (degrees)	Spread
1	Communications	55	650	53	24	random
2	Communications	35	700	45	125	random
3	Communications	100	900	70	67	even
4	Communications	100	900	70	231	even
5	Processing	20	525	45	65	random
6	Processing	45	375	97	44	random
7	Remote Sensing (type 1)	50	550	98	133	random
8	Remote Sensing (type 1)	30	350	45	90	random
9	Remote Sensing (type 2)	30	350	45	110	even
10	Remote Sensing (type 2)	75	725	98	12	even

8.2 Simulation Performance Metrics

A numerical score is assigned for specific actions taken by each autonomous spacecraft agent. The scores are multiplied by the duration of each action and accumulated over the entire simulated duration. The general success criteria for the remote sensing agents include:

- activation of sensors when the target is in view,
- opportunistic observations of secondary targets,
- transferring data to the communication support nodes, and
- maintaining baseline readiness for future tasking.

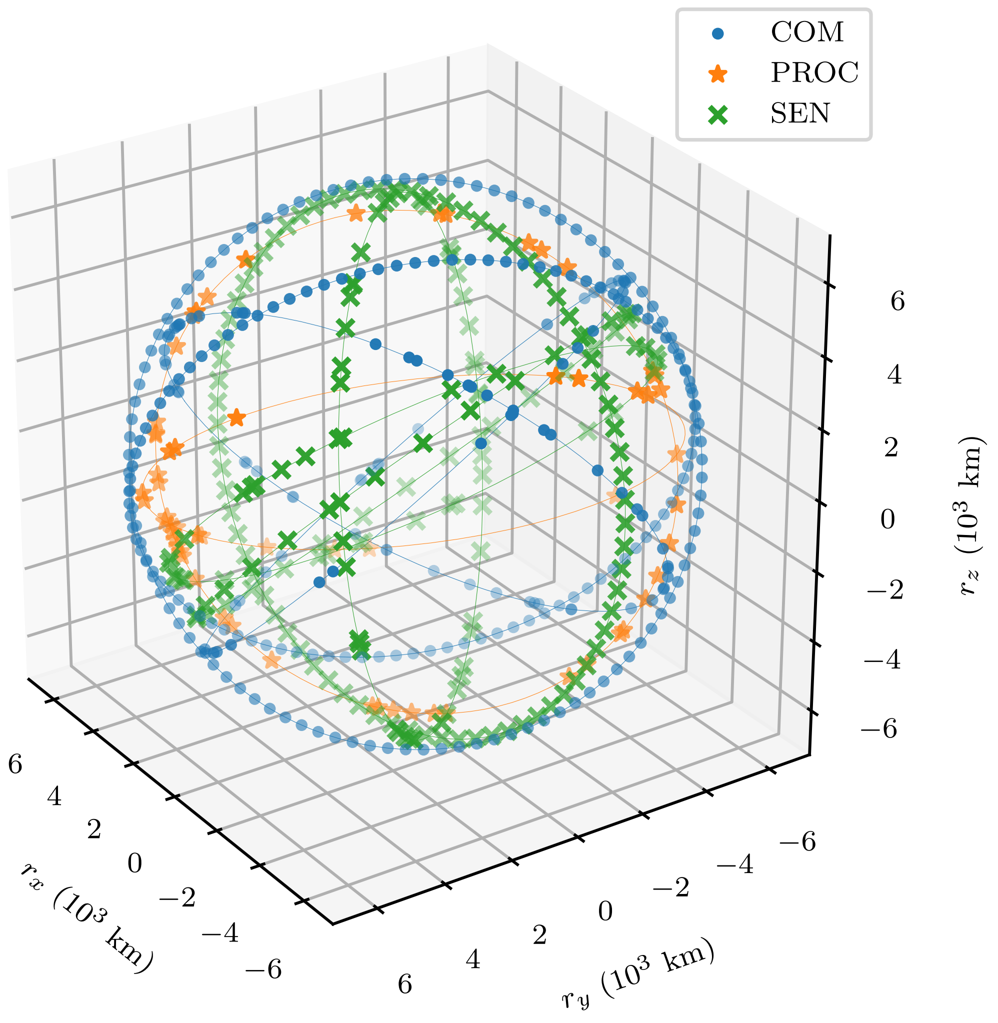


Figure 8.1 Distribution of all spacecraft in Earth-centered inertial coordinates

A score of zero is assigned for the entire simulation duration if the spacecraft battery is ever fully depleted.

8.2.1 Spacecraft Performance

Each sensor earns a different score rate when activated. Bonus scoring is awarded for activating sensors in combination. Tables 8.2 and 8.3 show the power resource consumption rates and the scores for each combination of sensor devices active for the first and second types. A score multiplier is applied when sensors are activated in sight of the tasking target. Thus, to maximize score, the autonomous agent must activate its entire payload package

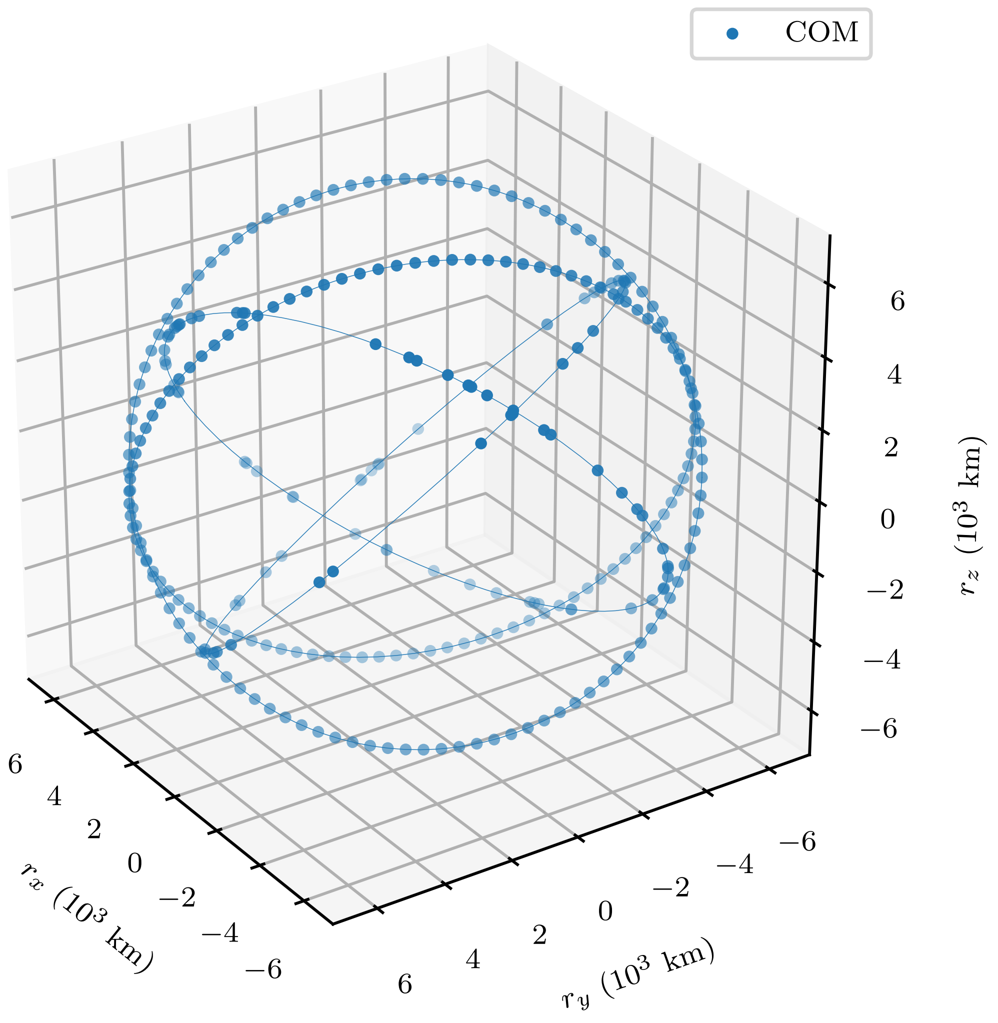


Figure 8.2 Distribution of communications spacecraft in Earth-centered inertial coordinates

when in view of the target. Each satellite is also given a reward when it meets battery self-preservation metric of at least 25% capacity. The score is nullified for the duration of the simulation for any individual spacecraft if its battery ever drains to zero stored energy.

The performance of the network is measured by combining the aggregate score of each individual spacecraft. More complex metrics could be developed to account for latency of data delivery, propagation of new tasking, mitigating faults before they happen, or other actions that promote good network behavior. These are beyond the scope of this example

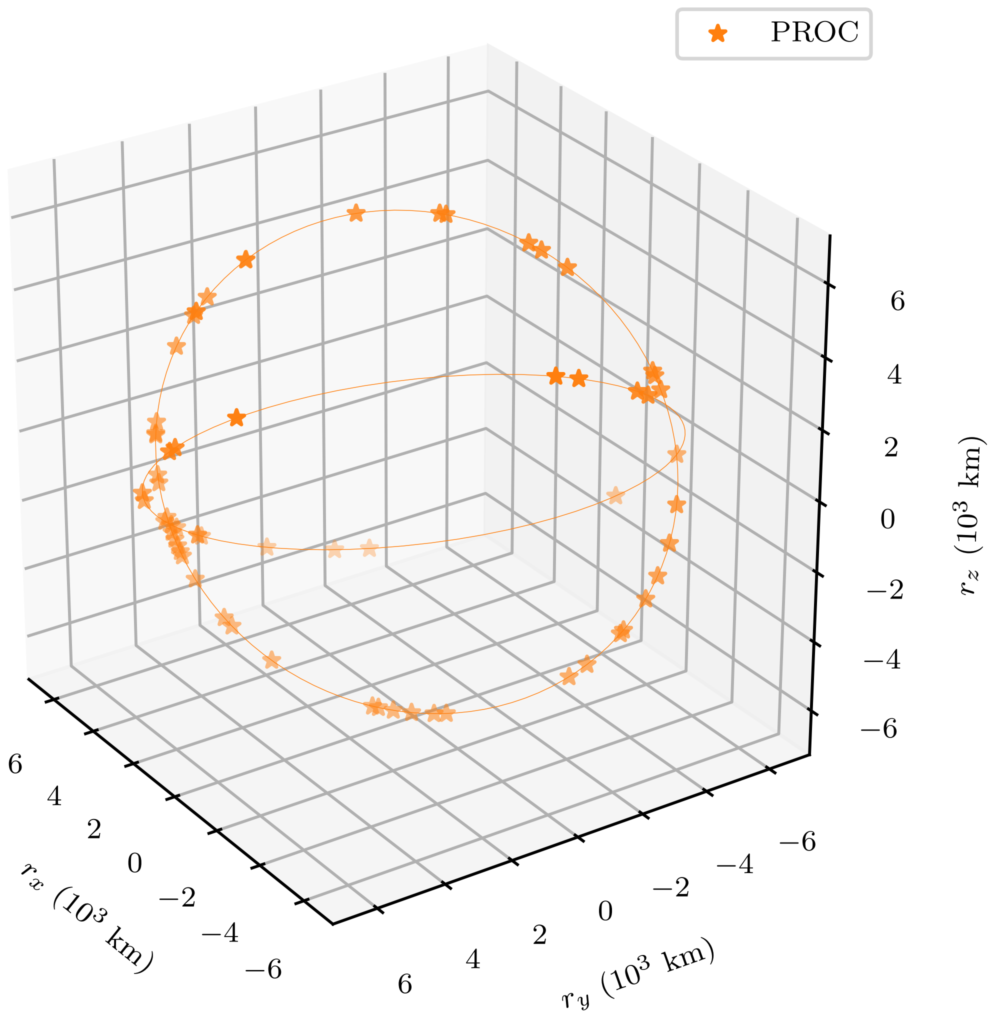


Figure 8.3 Distribution of processing spacecraft in Earth-centered inertial coordinates

problem as the focus is on showcasing the decision making system and not communications dynamics.

8.3 HEADS System Design

This section provides an overview of the HEADS system design process as applied to the remote sensor spacecraft NDSS problem. First, the signals of interest are defined with focus on operating the sensor payloads. Each output signal is mapped to a single ensemble of experts where an individual expert focuses on a specific aspect of the problem. The rules driving the logic for each expert are then defined followed by fuzzy set parameterizations

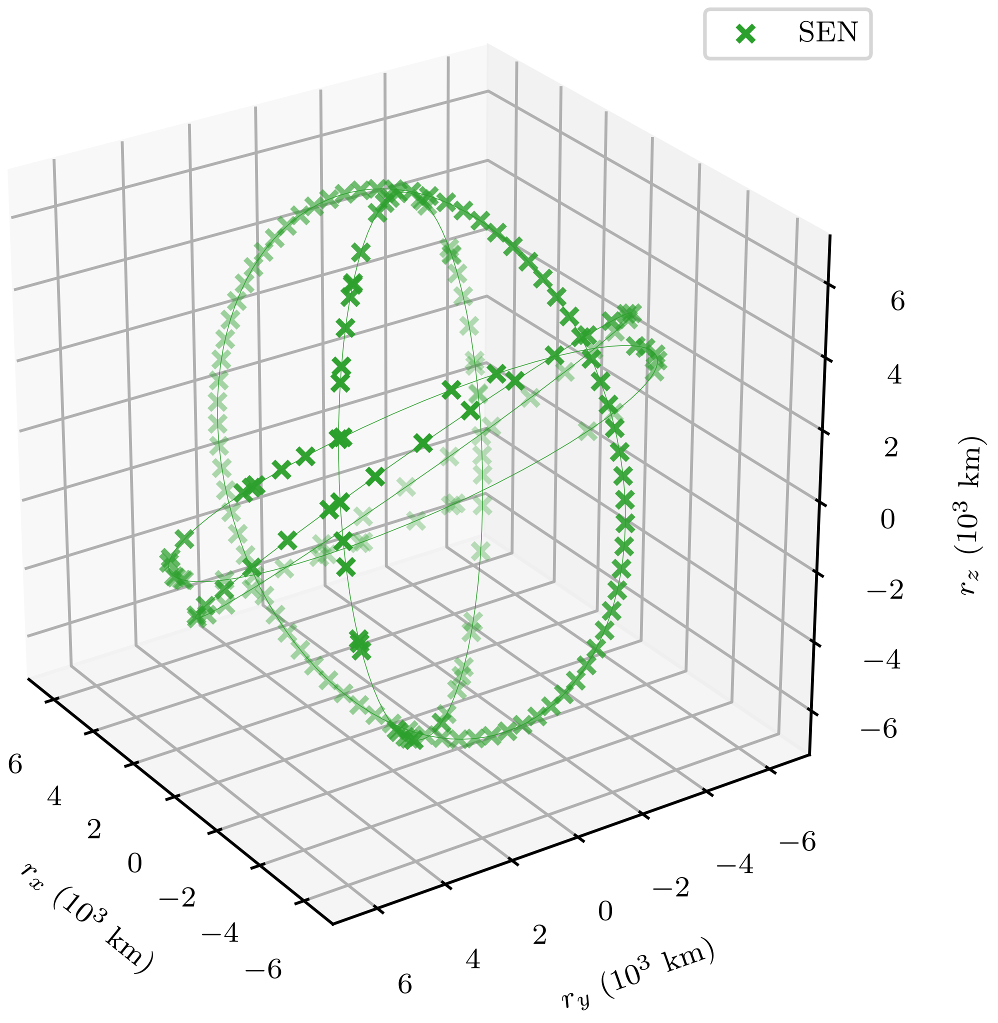


Figure 8.4 Distribution of remote sensing spacecraft in Earth-centered inertial coordinates

for each rule antecedent. Finally, the design introspection analysis metrics are provided for the system. As a simplifying measure the same HEADS system design is used for all sensor nodes in the network.

8.3.1 Signal Identification

The operation of the sensor payloads onboard the sensor nodes of the network is the primary objective. Of particular interest is operation of the sensors when a ground target is in view. However, operating the sensors should not put the spacecraft into a negative state.

Table 8.2 Remote sensing resource use and scoring (type 1)

Active Sensor/Device					
WVOC	NVOC	NIRC	DTX	Power (W)	Score (points per minute)
None (baseline/idle)				3	0
X				4	1
	X			5	1
		X		7	2
			X	12	3
X	X			9	3
X		X		11	4
X			X	16	5
	X	X		12	5
	X		X	17	4
		X	X	19	6
X	X	X		16	6
X	X		X	21	7
X		X	X	23	7
	X	X	X	24	8
X	X	X	X	28	10

The input signals available to a spacecraft typically include onboard telemetry such as battery power and whether the vehicle is in the sun. Other information can be derived through cross-linking with neighboring spacecraft or fault identification subsystems. Since observing a ground target is important for this simulated mission it is assumed the spacecraft have onboard positioning information and can determine the amount of time until the target is in view. Output signals include intermediate (self good) and terminal (sensor activation, crosslink radio activation, fault recovery) signals. Table 8.4 details the input and output signals, minimum and maximum expected values, and a nominal value used for sensitivity analysis.

The inputs and outputs span multiple categories of information including power, target visibility, neighbor counts and conditions, and fault mitigation. Each of these categories provides the scope for expert reasoning. It is possible for a logic expert to include multiple correlated categories within its ruleset, however this may lead to increased overall complexity.

8.3.2 Rule Base Definition

The following describes a ruleset from the perspective of assessing spacecraft health status in the context of power signals:

Table 8.3 Remote sensing resource use and scoring (type 2)

Active Sensor/Device					
WVOC	LALT	SAR	DTX	Power (W)	Score (points per minute)
None (baseline/idle)				5	0
X				4	1
	X			17	3
		X		38	8
			X	12	3
X	X			21	5
X		X		42	10
X			X	16	5
	X	X		55	12
	X		X	29	7
		X	X	50	13
X	X	X		59	15
X	X		X	54	10
X		X	X	33	15
	X	X	X	67	18
X	X	X	X	71	20

Table 8.4 Input and output signals for sensor node logic

Var.	Signal Name	In/Out	Initial Value	Min Value	Max Value	Nominal Value
x_1	stored_energy	input	random	0.0	100.0	80.0
x_2	sunlit	input	orbit-dependent	False	True	True
x_3	dt_sun_s	input	orbit-dependent	0.0	2100.0	
x_4	target_visible	input	orbit-dependent	False	True	True
x_5	dt_target_s	input	orbit-dependent	0.0	86400.0	0.0
x_6	COM_neighbors_count	input	orbit-dependent	0	100	1
x_7	PROC_neighbors_count	input	orbit-dependent	0	100	1
x_8	SEN_neighbors_count	input	orbit-dependent	0	100	1
x_9	SEN_neighbors_good	input	orbit-dependent	0	100	1
x_{10}	SEN_neighbors_bad	input	orbit-dependent	0	100	0
x_{11}	fault_detected	input	random	False	True	False
y_1	self_good	output	1.0	0.0	1.0	1.0
y_2	sensor_1_active	output	0.0	0.0	1.0	1.0
y_3	sensor_2_active	output	0.0	0.0	1.0	1.0
y_4	sensor_3_active	output	0.0	0.0	1.0	1.0
y_5	dtx_active	output	0.0	0.0	1.0	1.0
y_6	fault_recovery	output	0.0	0.0	1.0	0.0

1. If battery is critically low THEN self is bad
2. If battery is low AND time to sun is long THEN self is bad
3. If battery is low AND (time to sun is short OR in the sun) THEN self is marginal
4. IF battery is good AND time to sun is long THEN self is marginal
5. If battery is good AND (time to sun is short OR in the sun) THEN self is good
6. If battery is full THEN self is good

Table 8.5 provides the HEADS-design linguistic rules. The rules prime fuzzy sets for both inputs and outputs linked to signals as described in Table 8.6. Appendix A shows the rules and fuzzy sets for the rest of the HEADS system design. Writing rules with few antecedent terms and minimal parenthetical groupings ensures readability and understandability. Reducing rule complexity supports sustainable logic and leads to better explainability during system introspection.

Table 8.5 Rules for expert $s_{1,1}$ of ensemble y_1

Rule Index	Rule
r_1_1_01	batt_critical THEN self_bad
r_1_1_02	batt_low AND dt_sun_long THEN self_bad
r_1_1_03	batt_low AND (dt_sun_short OR sunlit) THEN self_marginal
r_1_1_04	batt_good AND dt_sun_long THEN self_marginal
r_1_1_05	batt_good AND (dt_sun_short OR sunlit) THEN self_good
r_1_1_06	batt_full THEN self_good

8.3.3 Fuzzy Set Parameterization

Defining the fuzzy set parameters completes the HEADS system. These parameters ensure each fuzzy set activates to the correct degree over its signal domain. Fuzzy sets may overlap as a way to capture uncertainty in degree of activation. Table 8.7 details the parameters given to the fuzzy sets described in Table 8.6. The rest of the fuzzy set parameters used for this example problem are found in Appendix A. Figures 8.5-8.8 show the fuzzy set activation functions used in this example.

Table 8.6 Fuzzy sets for expert $s_{1,1}$ of ensemble y_1

Antecedent/ Consequent	Fuzzy Set	Signal
antecedent	batt_critical	x_1: stored_energy
antecedent	batt_low	x_1: stored_energy
antecedent	batt_good	x_1: stored_energy
antecedent	batt_full	x_1: stored_energy
antecedent	sunlit	x_2: sunlit
antecedent	dt_sun_short	x_3: dt_sun_s
antecedent	dt_sun_long	x_3: dt_sun_s
consequent	self_bad	y_1: self_good
consequent	self_marginal	y_1: self_good
consequent	self_good	y_1: self_good

Table 8.7 Fuzzy set parameters for expert $s_{1,1}$ of ensemble y_1

Antecedent/ Consequent	Fuzzy Set	Activation Function	Parameters
antecedent	batt_critical	LTRAP	[10, 20]
antecedent	batt_low	TRAP	[15, 20, 25, 30]
antecedent	batt_good	TRAP	[30, 50, 70, 80]
antecedent	batt_full	RTRAP	[60, 80]
antecedent	sunlit	BINARY	[]
antecedent	dt_sun_short	LTRAP	[300, 600]
antecedent	dt_sun_long	RTRAP	[300, 900]
consequent	self_bad	LTRAP	[0.4, 0.5]
consequent	self_marginal	TRAP	[0.45, 0.5, 0.55, 0.65]
consequent	self_good	RTRAP	[0.6, 0.7]

8.3.4 Design Introspection

Introspection during and after design informs the designer of overall completeness and complexity of the system, ensembles, and experts. The following discussion covers the introspection metrics presented in Chapter 6.

8.3.4.1 Binary Coverage

Binary coverage provides a high-level overview of mappings of signals throughout the system. Table 8.8 shows the system hierarchy and binary coverage at each level. Refer to Table 8.4 for mappings from input and output state variables to the signal names. A binary coverage '1' indicates that the signal is used by the system, ensemble, or expert as an input for logic.

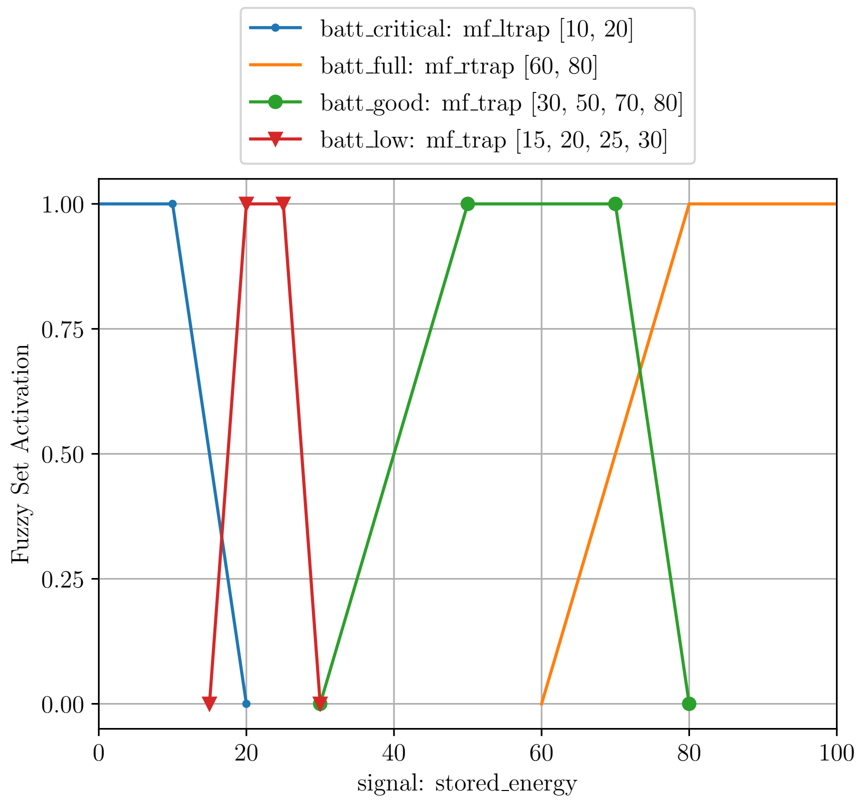


Figure 8.5 Fuzzy set parameters for ensemble y_1, expert $s_{1,1}$, input signal stored_energy

Note that the output signal y_5 has a '0' at the system level. This indicates that this signal, 'dtx_active,' is not used as an input for logic and is considered a terminal output. All of the other outputs are used as input to logic rules and are therefore intermediate outputs. Note also that there is no reason to not use an intermediate output as a control signal. This example uses each sensor activation output signal as an input to logic and to drive sensor payload activity.

8.3.4.2 Weighted Coverage

Weighted coverage provides a high-level overview of signal importance throughout the system. Table 8.9 shows the weighted coverage across the system hierarchy. Refer to Table 8.4 for mappings from input and output state variables to the signal names.

The weighted coverage value is a count for how many fuzzy sets use a signal. Larger values may indicate telemetry or subsystems that have larger impact on the autonomous

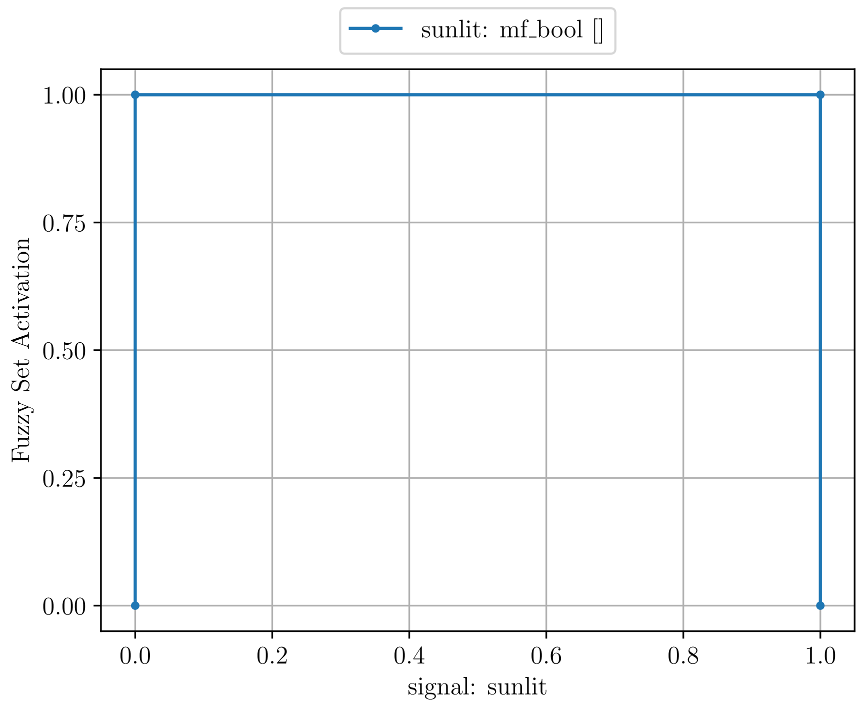


Figure 8.6 Fuzzy set parameters for ensemble y_1, expert $s_{1,1}$, input signal sunlit

reasoning. The system level weighted coverage is the sum over all ensemble weighted coverage vectors. Likewise, each ensemble weighted coverage it the sum over its expert's weighted coverage vectors.

8.3.4.3 Complexity Metrics

Complexity and understandability are inversely proportional. Measuring complexity is difficult and important as discussed in Chapters 3 and 4. This analysis is based on measuring system size, interconnectedness, and conflict through methods developed in Chapter 6 including:

- $\psi_{j,p,rules} \equiv$ the number of rules
- $\psi_{j,p,fsets} \equiv$ the number of fuzzy sets
- $\psi_{j,p,z_l} \equiv$ the number of fuzzy sets used as rule antecedents defined for a signal
- $\psi_{j,p,w,ANT} \equiv$ the number of antecedents for a rule (known as rule compactness)

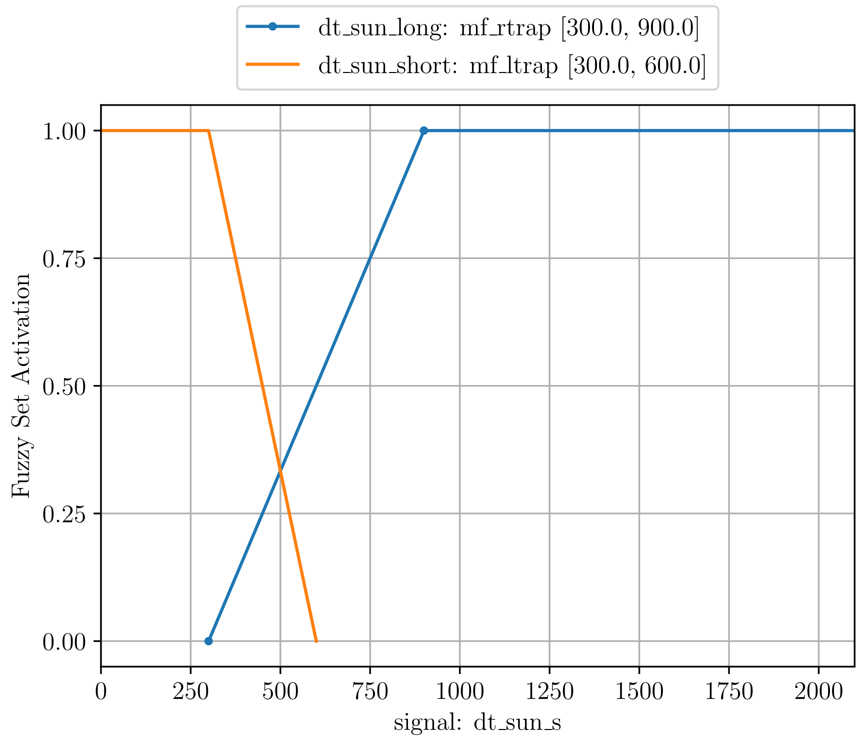


Figure 8.7 Fuzzy set parameters for ensemble y_1, expert $s_{1,1}$, input signal stored_energy

- $\psi_{j,p,ANT} \equiv$ the maximum value for rule compactness over all rules
- $\psi_{j,p,\overline{ANT}} \equiv$ the average value for rule compactness over all rules
- $\boldsymbol{\Psi}_{j,p} \equiv$ the similarity between each pair of rules in an expert system

Recall that y_j is used to define an ensemble, and $s_{j,p}$ an expert within an ensemble. The same subscript indices are used for the complexity metrics. Each metric can be extended upward through the hierarchy by taking the sum, maximum, or average across all constituent experts or ensembles. Table 8.10 shows the rules, fuzzy sets, and rule compactness metrics for the system, first ensemble, and both experts in the ensemble. Table 8.11 presents the count of fuzzy sets defined using each signal. Tables 8.12-8.13 shows the rule similarity matrices for experts $s_{1,1}$ and $s_{1,2}$ respectively. Complete system, ensemble, and expert complexity metrics are in Appendix B.

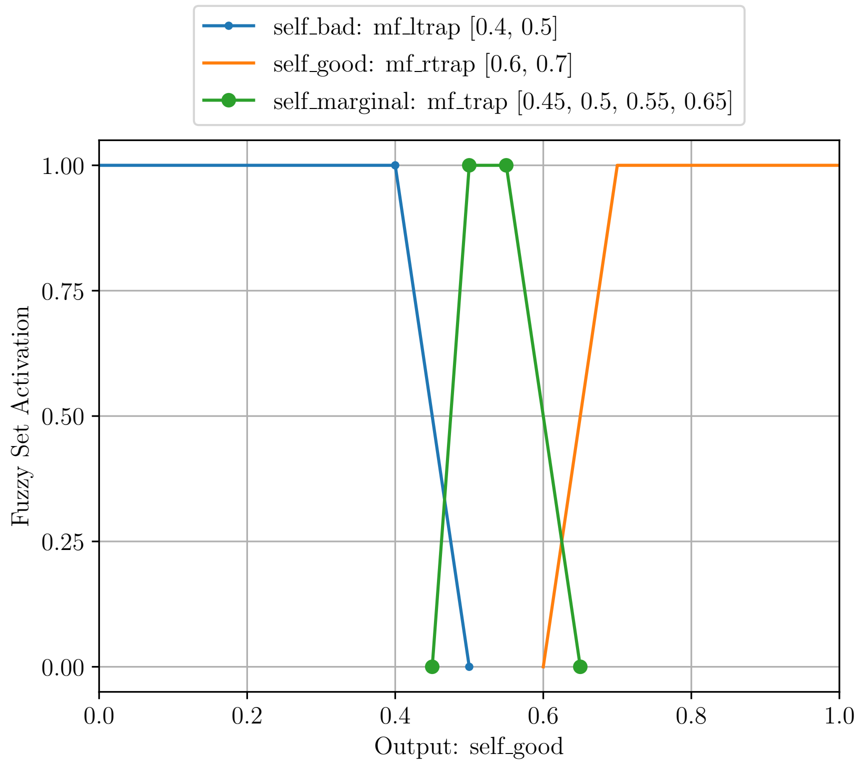


Figure 8.8 Fuzzy set parameters for ensemble y_1, expert $s_{1,1}$, output signal self_good

8.3.4.4 Sensitivity Analysis

The final introspective analysis to perform is measuring output signal sensitivity to changes in the signals used by antecedent fuzzy sets. The approach presented here assumed nominal values for each signal (as shown in Table 8.4) and varied the signals used by each expert. The output fuzzy set activation and confidence are plotted as a function of the varied parameter. Figures 8.9-8.11 show sensitivity of expert $s_{1,2}$ focused on determining the 'self_good' signal based on fault-related input signals. For this expert, larger output activation correlates with better health assessment.

Note that the nominal values used across the rest of the signal space may have a large impact on the resulting sensitivity analysis. This can be seen in Figure 8.9 where the output activation does not change regardless of the amount of stored energy. Physically this means the expert would self-assess health as good even with a dead battery. However, diving

Table 8.8 Example HEADS binary coverage

Hierarchy	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	y_1	y_2	y_3	y_4	y_5	y_6
System	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
y_1	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1
s_1_1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
s_1_2	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
y_2	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	0	1
s_2_1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
s_2_2	0	0	0	1	1	0	0	0	0	0	0	0	1	1	1	0	0
s_2_3	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0
s_2_4	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1
y_3	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	0	1
s_3_1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
s_3_2	0	0	0	1	1	0	0	0	0	0	0	0	1	1	1	0	0
s_3_3	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0
s_3_4	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1
y_4	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	0	1
s_4_1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
s_4_2	0	0	0	1	1	0	0	0	0	0	0	0	1	1	1	0	0
s_4_3	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0
s_4_4	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1
y_5	1	1	1	0	0	1	1	0	0	0	1	1	1	1	1	0	1
s_5_1	1	1	1	0	0	1	1	0	0	0	0	0	1	1	1	0	0
s_5_2	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1
y_6	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
s_6_1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1

into the rules for this expert (see Table A.1) the battery signal must correlate with the fault signal for the rule to activate. This shows how having more than one set of 'nominal' inputs used for sensitivity analysis can be valuable.

Figure 8.10 shows 'self good' sensitivity to the boolean 'fault detected' signal. The self good output (for this expert) is driven to zero if a fault is detected when all other signals are nominal. The 'fault recovery' output signal would be used to drive the system into fault mitigation modes. As shown in Figure 8.11 the 'self good' output is reduced if the fault recovery output goes above its activation level (0.5). Note however that the self good output is still above its activation level. This could be due to other nominal signals keeping the output high or it could indicate the need for rule modification or fuzzy set tuning.

Table 8.9 Example HEADS weighted coverage

Hierarchy	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	y_1	y_2	y_3	y_4	y_5	y_6
System	26	9	12	6	6	4	4	6	6	6	14	4	4	4	4	0	8
y_1	8	2	4	0	0	0	0	0	0	0	3	0	0	0	0	0	1
s_1_1	6	2	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
s_1_2	2	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	1
y_2	4	2	2	2	2	0	0	2	2	2	2	1	1	1	1	0	1
s_2_1	4	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
s_2_2	0	0	0	2	2	0	0	0	0	0	0	0	1	1	1	0	0
s_2_3	0	0	0	0	0	0	0	2	2	2	0	0	0	0	0	0	0
s_2_4	0	0	0	0	0	0	0	0	0	0	2	1	0	0	0	0	1
y_3	4	2	2	2	2	0	0	2	2	2	2	1	1	1	1	0	1
s_3_1	4	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
s_3_2	0	0	0	2	2	0	0	0	0	0	0	0	1	1	1	0	0
s_3_3	0	0	0	0	0	0	0	2	2	2	0	0	0	0	0	0	0
s_3_4	0	0	0	0	0	0	0	0	0	0	2	1	0	0	0	0	1
y_4	4	2	2	2	2	0	0	2	2	2	2	1	1	1	1	0	1
s_4_1	4	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
s_4_2	0	0	0	2	2	0	0	0	0	0	0	0	1	1	1	0	0
s_4_3	0	0	0	0	0	0	0	2	2	2	0	0	0	0	0	0	0
s_4_4	0	0	0	0	0	0	0	0	0	0	2	1	0	0	0	0	1
y_5	4	1	2	0	0	4	4	0	0	0	2	1	1	1	1	0	2
s_5_1	4	1	2	0	0	4	4	0	0	0	0	0	1	1	1	0	0
s_5_2	0	0	0	0	0	0	0	0	0	0	2	1	0	0	0	0	2
y_6	2	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	2
s_6_1	2	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	2

Table 8.10 Example HEADS complexity metrics for rules and fuzzy sets for ensemble y_1

Hierarchy	ψ_{rules}	ψ_{fsets}	$\psi_{w,ANT}$	ψ_{ANT}	$\psi_{\overline{ANT}}$
System					
y_1	10	11		3	1.75
s_1_1	6	7		3	2.0
r_1_1_01			1		
r_1_1_02			2		
r_1_1_03			3		
r_1_1_04			2		
r_1_1_05			3		
r_1_1_06			1		
s_1_2	4	4		2	1.5
r_1_2_01			2		
r_1_2_02			2		
r_1_2_03			1		
r_1_2_04			1		

Table 8.11 Example HEADS fuzzy set to signal mapping complexity for ensemble y_1

Hierarchy	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	y_1	y_2	y_3	y_4	y_5	y_6
System	18	5	10	3	6	1	1	6	6	6	6	4	4	4	4	0	6
y_1	6	1	2	0	0	0	0	0	0	0	1	0	0	0	0	0	1
s_1_1	4	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
s_1_2	2	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1

Table 8.12 Example HEADS rule similarity matrix for expert $s_{1,1}$

Rule	r_1_1_01	r_1_1_02	r_1_1_03	r_1_1_04	r_1_1_05	r_1_1_06
r_1_1_01	1.0	0.0	0.0	0.0	0.0	0.0
r_1_1_02	0.0	1.0	0.1	0.1	0.0	0.0
r_1_1_03	0.0	0.1	1.0	0.0	0.2	0.0
r_1_1_04	0.0	0.1	0.0	1.0	0.1	0.0
r_1_1_05	0.0	0.0	0.2	0.1	1.0	0.0
r_1_1_06	0.0	0.0	0.0	0.0	0.0	1.0

Table 8.13 Example HEADS rule similarity matrix for expert $s_{1,2}$

Rule	r_1_2_01	r_1_2_02	r_1_2_03	r_1_2_04
r_1_2_01	1.0	0.14	0.14	0.0
r_1_2_02	0.14	1.0	0.14	0.0
r_1_2_03	0.14	0.14	1.0	0.0
r_1_2_04	0.0	0.0	0.0	1.0

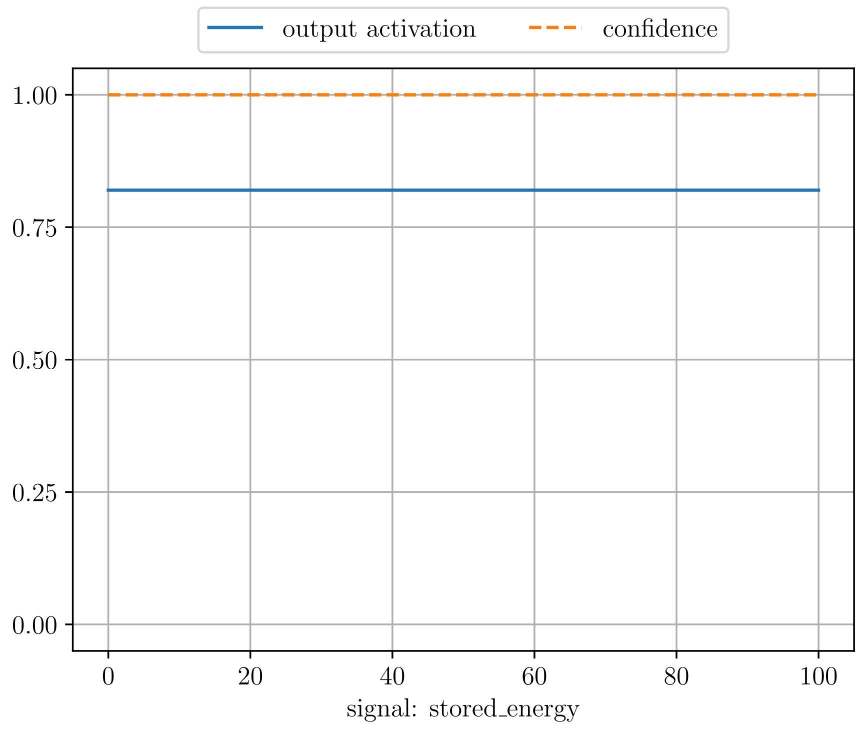


Figure 8.9 Example HEADS sensitivity for ensemble y_1 expert $s_{1,2}$, output 'self_good', variable signal x_1 ('stored_energy')

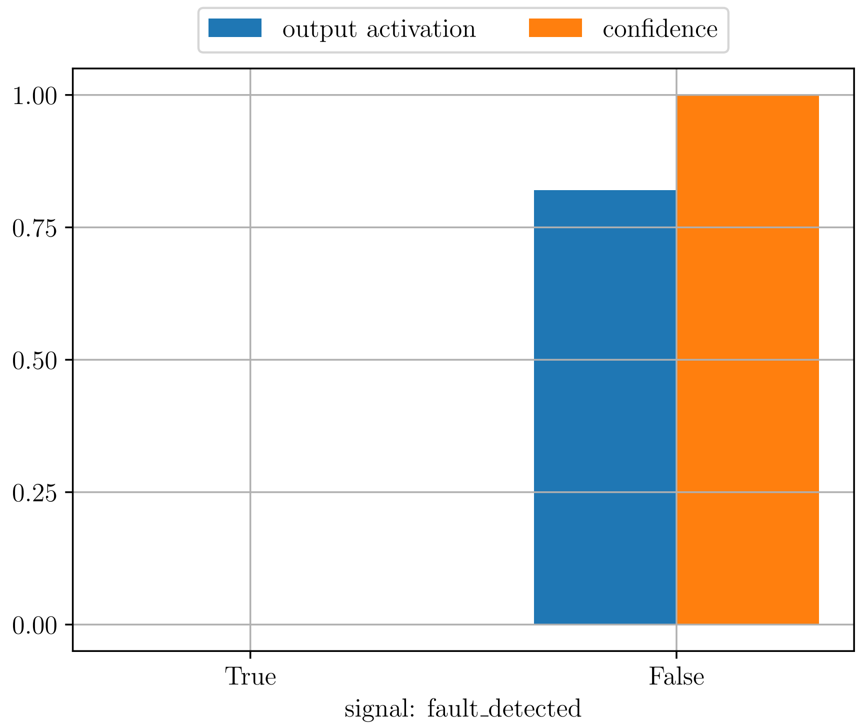


Figure 8.10 Example HEADS sensitivity for ensemble y_1 expert $s_{1,2}$, output 'self_good', variable signal x_{11} ('fault_detected')

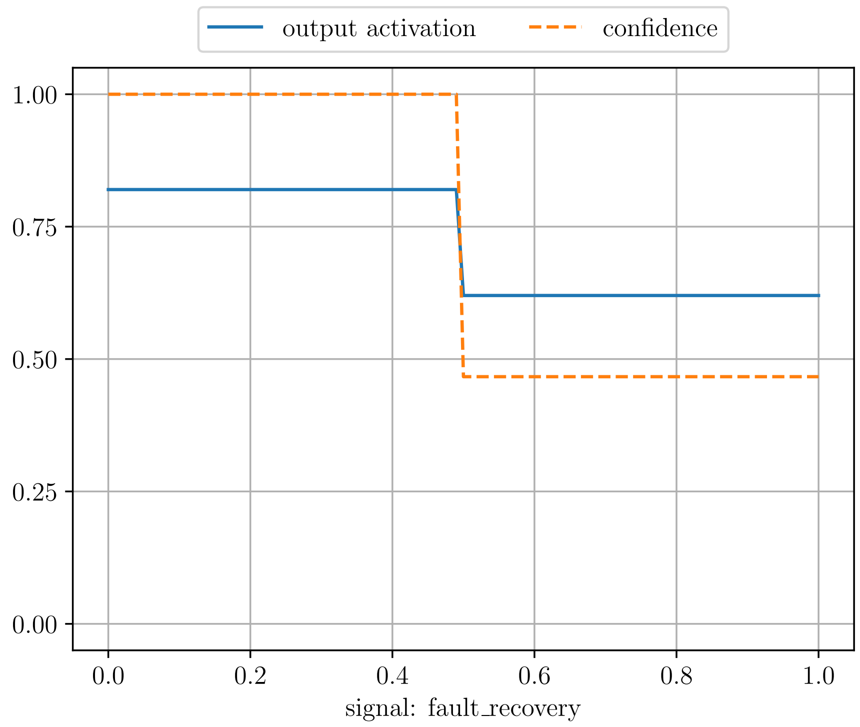


Figure 8.11 Example HEADS sensitivity for ensemble y_1 expert $s_{1,2}$, output 'self_good', variable signal y_6 ('fault_recovery')

Chapter 9: Simulation Results and Performance Analysis

This section provides an overview of the performance for the as-designed system. At this point no post-simulation analysis or tuning was done to adapt the rules or the fuzzy set parameters. Metrics are presented for all sensor nodes at a high level with detailed inspection of specific nodes to show the HEADS system at work. Special attention is given to the score-based performance metrics chosen for this example including maintaining battery status and activating sensors when the ground target is in view.

The simulation for this example problem was done in two parts:

1. the orbital dynamics were simulated for all satellites and orbit-dependent state vectors computed, and

2. the HEADS system was initialized for all sensor nodes in the network and the reasoning simulation done to compute dynamic input and output states.

The scores were computed during post-processing.

The results for this pre-tuning simulation show the general success of the HEADS system in operating the sensor network. Most of the sensor nodes maintain positive power status. Many nodes activate their sensor payloads when the target is in view and use spare power to make opportunistic observations when the target is not visible. A few possible improvements are easy to identify relating to fuzzy set activations given known system states. These issues are simple to mitigate through rule analysis and fuzzy set adaptation. Such a manual tuning approach is taken with results shown in Chapter 10.

A full complement of system states and outputs and a selection of fuzzy set and rule activations are shown in detail in Appendix C.

9.1 Sensor Constellation Performance

There are 185 sensor nodes in the simulated network of satellites. These nodes are split between four unique orbits as specified in Table 8.1. Two types of spacecraft are simulated with the primary difference between types being the power required to run each sensor and the resulting performance score when sensors are active. All nodes are assumed to have the same amount of energy storage. Scoring is detailed in Table 8.2 and Table 8.3 for nodes of type 1 and type 2 respectively.

The sensor nodes are expected to activate their sensors when the ground target is in view and maintain a minimum level of stored energy to ensure good health and readiness.

9.1.1 State: Energy and Power

The stored energy state impacts power-related decision making. Figure 9.1 shows signal x_1 for stored energy with detail shown in Figure 9.2. Of note here are the two main equilibria which correspond to the sensor node type 1 (higher) and type 2 (lower) average stored energy.

A small but nonzero number of nodes (18 of 185) have stored energy that drops to zero at least once. Only 5 nodes have zero stored energy for more than half of the simulated time. These are considered failures as the ruleset should ensure the system deactivates to prevent total power loss. However, as a non-tuned implementation of the HEADS system, the ruleset achieves the power management goal for 90% of the sensor network. This shows that the HEADS framework supports achieving good results without analysis and tuning.

The measure of power differential (where negative means drain and positive means generation) is a derived metric not used as a system signal but useful for design validation. In general, maximal utility drives power differential to zero. If a system generates more power than it uses, it could adapt its use profile to put the positive balance to work. If a system loses power it must adapt use to ensure total stored energy does not reach zero. Figure 9.3 and Figure 9.4 show the sensor network power differential. The initial negative differential

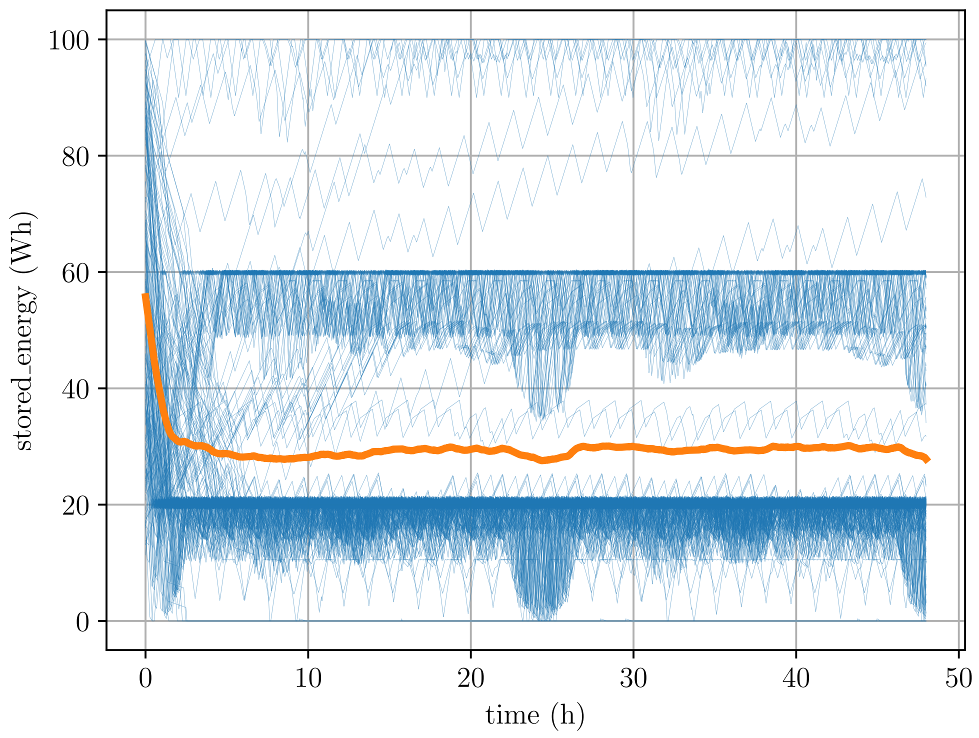


Figure 9.1   Sensor node performance: stored energy state

indicates that the sensor network had an excess of stored energy. This excess is quickly used up and the network achieves steady-state around zero power differential as expected.

9.1.2 State: Target Visible

Whether the target is visible depends on a satellite's orbit. The target is modeled as a single ground location at moderate latitude (20°S). The primary objective of the sensor network is to observe the ground target

Visibility matters for the sensor nodes in the network as it drives the need to activate sensor payloads. Target visibility across the network depends on distribution of orbits and satellites within the orbits. Figure 9.5 shows target visibility over the entire simulation while Figure 9.6 is a detail view.

The target is in view of at least one sensor node a majority (85.8%) of the simulated time. This metric is useful for orbit design validation. Target visibility can also be a driver for system and rule design. If the target is often visible perhaps the sensors require less

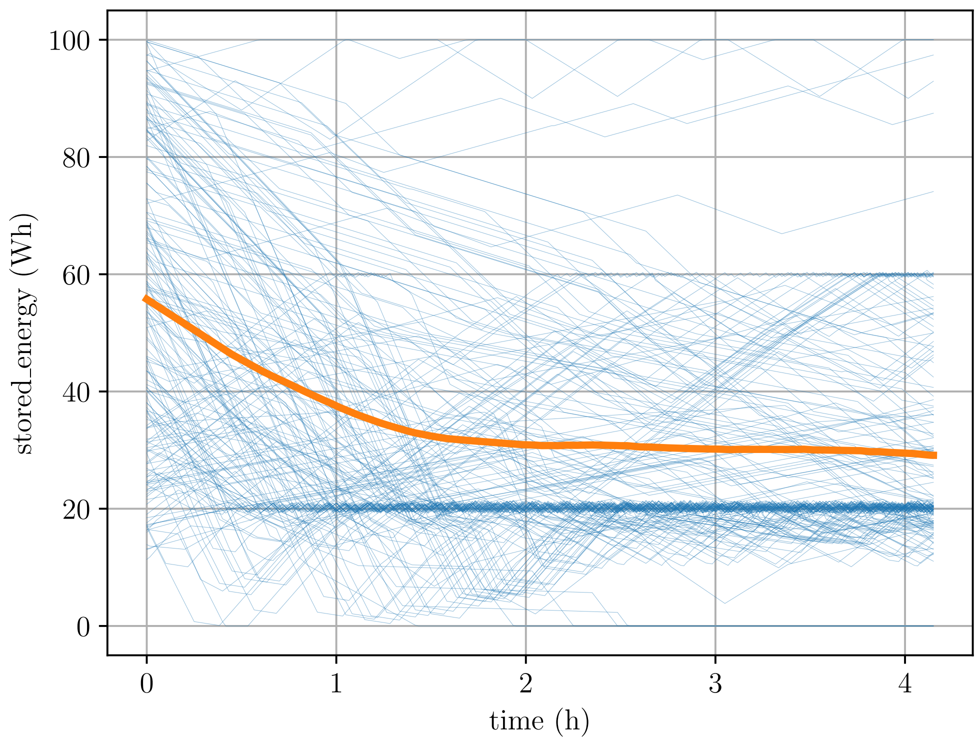


Figure 9.2 Sensor node performance: stored energy state (detail of first four hours)

performance as temporal resolution would mitigate spatial resolution deficiency. From an operational perspective target visibility informs expected per-spacecraft activity. If a node has visibility often it may require more energy storage or power generation to meet sensor demands. The rule designer may also wish to tune performance to not drive the system into poor health.

9.1.3 State: Fault Detected

This simulation includes random faults applied to each node in the network. The implementation is stochastic where a small sample (5%) of all nodes are made 'faulty' with a majority of their time (99.9%) in a fault-detected state. Non-faulty nodes also suffer random fault-detected instances about 0.1% of the time. The simulated fault mechanism is quadrupled power drain for any idle power or activated subsystem. Figure 9.7 shows the 'fault-detected' state for a faulty node and a non-faulty node.

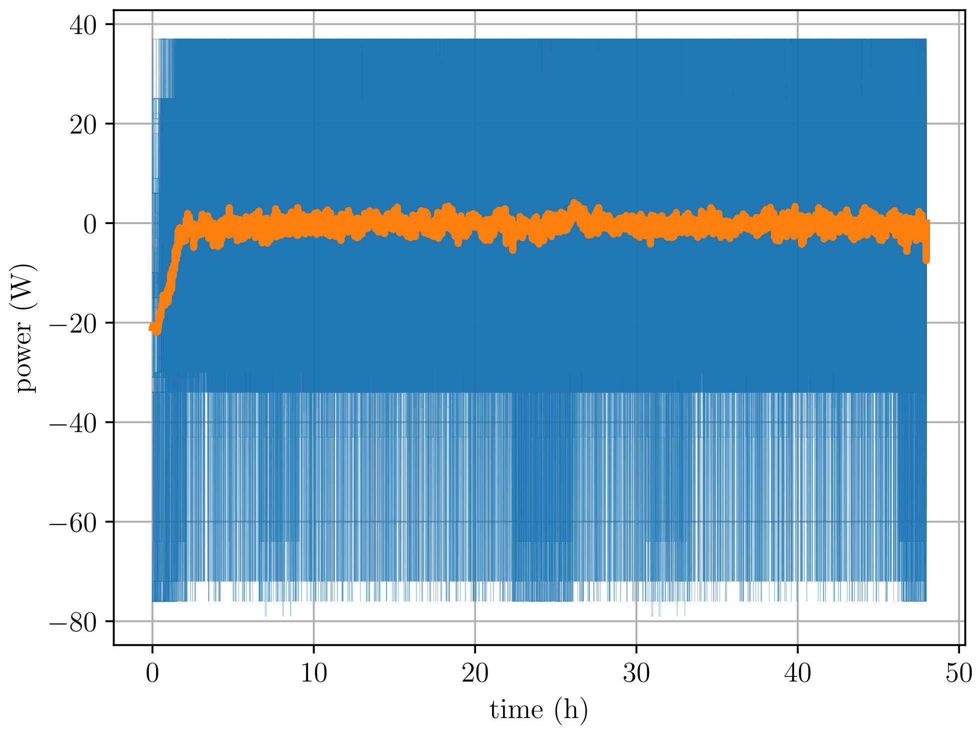


Figure 9.3 Sensor node performance: power differential

The expected system response to a detected fault is to enter into 'fault recovery' mode (output y_6) which discourages all sensor and high-bandwidth communications. The intent is to support a power-positive state to enable the system to perform fault mitigation activities and accept assistance from ground operators as needed.

9.2 Performance Detail

Two non-faulty nodes and two faulty nodes were selected to present individual node performance. Node B5 and node D34 are from type 1 and type 2 sensor families, respectively and represent nominal performance. Node A6 (type 1) and C28 (type 2) are simulated as faulty. This section provides a glimpse at input and output signals important to decision making. Output signals that drive performance scoring are examined in detail. Fuzzy set and rule activations are analyzed and provide further context for the selected actions.

The selected sensor node scores are compared to their orbit/type companions and the sensor network as a whole in Table 9.1. The table shows general statistics for the orbits

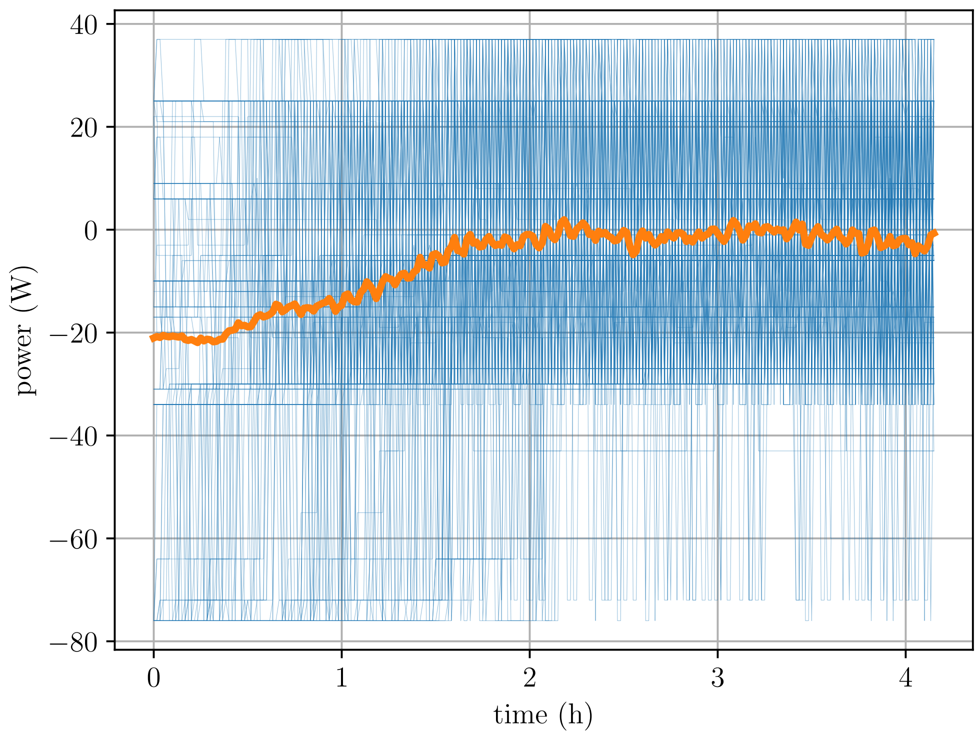


Figure 9.4 Sensor node performance: power differential (detail of first four hours)

including the average score by category. Maximum and minimum-nonzero scores are shown for the orbit. The maximum or minimum total, subtotal, self_good, and stored_energy scores A zero score was awarded to any satellite that had its stored energy go to zero at any time. A few things stand out as notable:

- the faulty nodes (A6 and C28) have total score close to each node's orbit average,
- sensor scoring (the Subtotal value) accounts for a majority of scoring for the healthy nodes (B5 and D34), and
- the different orbits and different satellite types provide different average value to the sensor network.

Scores and per-orbit statistics for the entire baseline simulation sensor network are in Table C.2 in Appendix C.

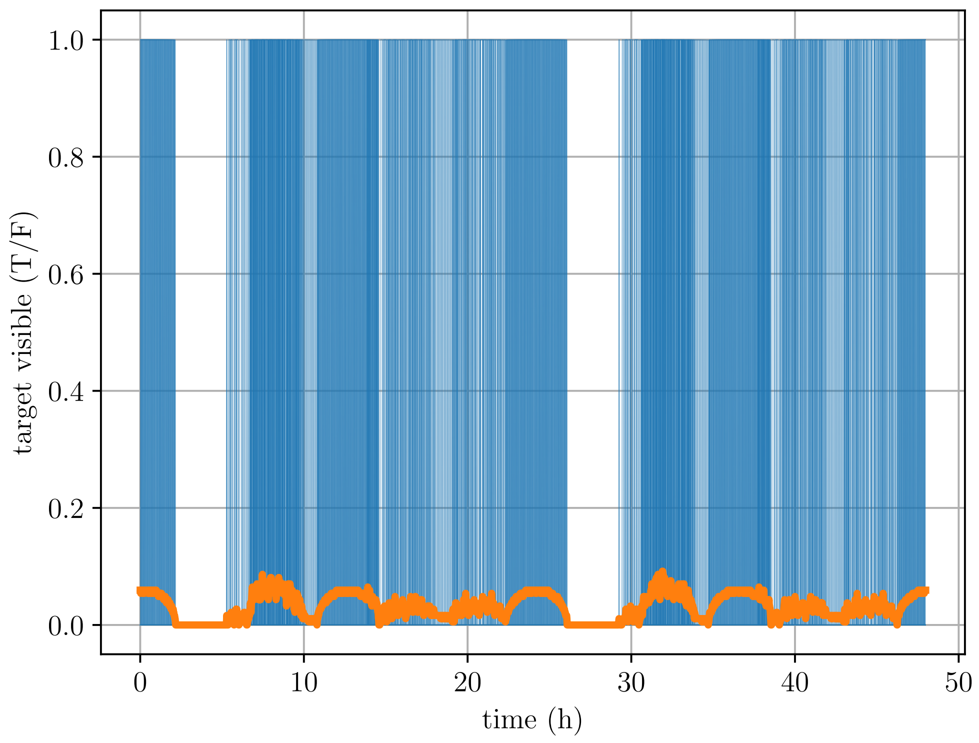


Figure 9.5 Sensor node performance: target visible state

9.2.1 Node B5 - Type 1 - Nominal

The B5 node is part of the type 1 sensor group meaning it has optical camera payloads and generates solar power at 21 Watts. The B orbit is relatively low (350 km) and at mid-inclination (45°). Its low altitude makes sending data to the communications nodes impossible, since all COM satellites are at altitudes of 650 km or higher and the high-bandwidth transmitter (DTX) payload works at range up to 250 km. Sensors in orbit B can pass data to the processor satellite subnetwork. The middle inclination provides good visibility to the ground target at -20.55°latitude.

Figure 9.8 shows when each sensor and high-bandwidth communications payload are active and scoring points for activity. The spikes in sensor activation correspond to ground target visibility shown in Figure 9.9. This indicates the ruleset activates all sensor payloads when the target is visible which is desired behavior. Sensor 3 is active for much more time than when the target is in view which proves that the node is capable of and

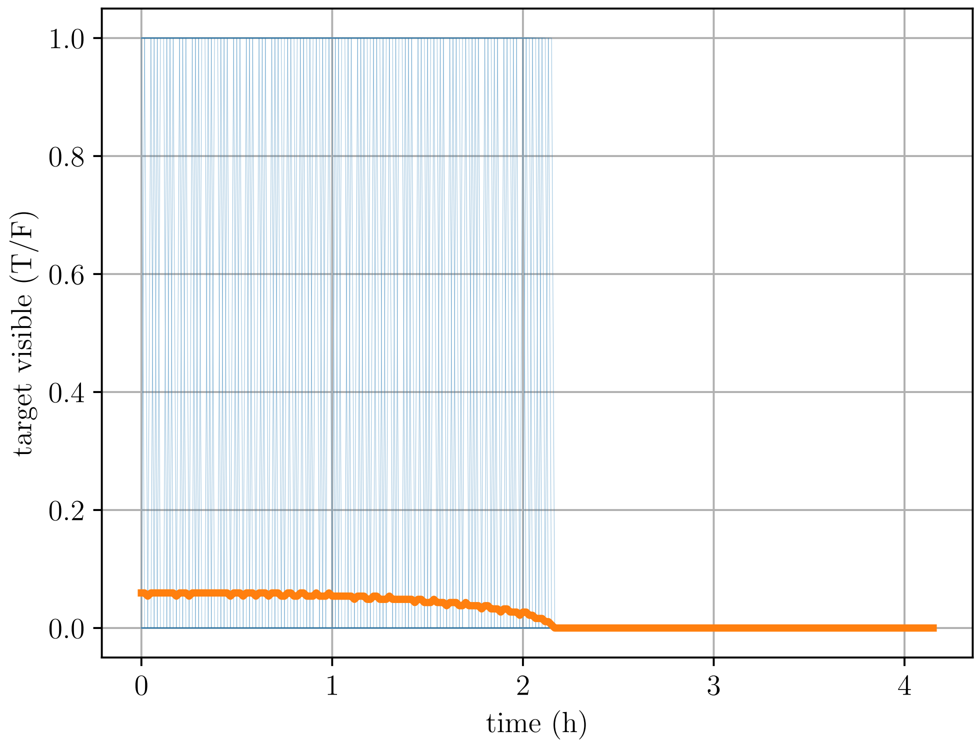


Figure 9.6 Sensor node performance: target visible state (detail of first four hours)

performs opportunistic observations. Note that the data transmission activates often. When activated with sensor 3, the entire system is in a power-negative state. Since there are no communication nodes in range and the processor nodes are not so consistently in range (see Figure 9.10) this is effectively wasting energy.

Why does the DTX payload activate so often? Inspection of the fuzzy sets and rules that activate provide the necessary insight to answer this question. There are two experts in the ensemble that drives DTX activation:

- s_5_1 focuses on power and sensor activation signals, and
- s_5_2 focuses on fault signals.

As shown in Figure 9.11, rule r_5_1_01 (batt_bad THEN dtx_off) dominates while all other rules never activate. Figure 9.12 shows the other expert's rule activation, with rule r_5_2_02 (NOT faulty AND NOT fault_recovery THEN dtx_on) always active and rule r_5_2_03 (NOT

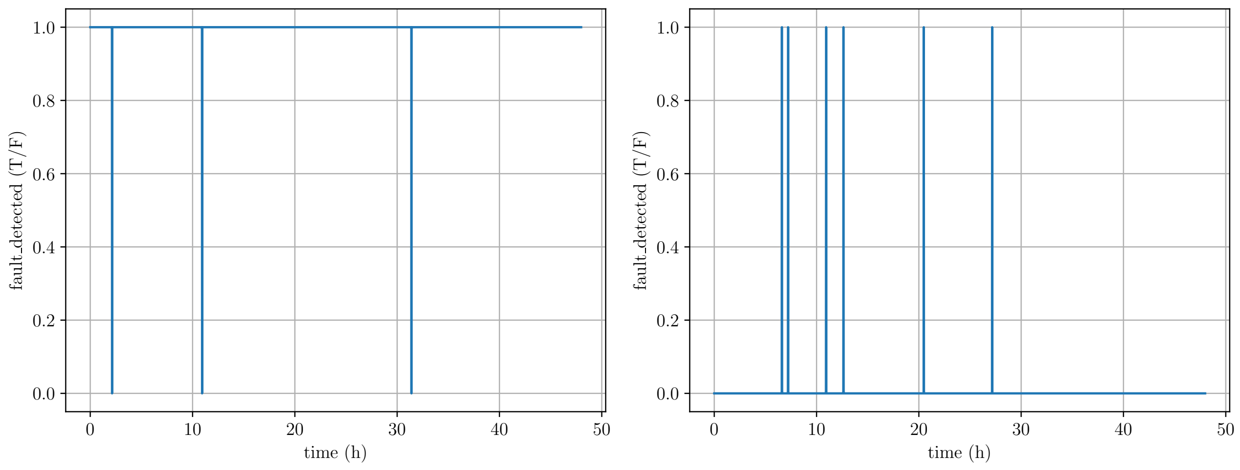


Figure 9.7 Faulty node (A6) and non-faulty node (B5) fault-detected state comparison

self_good THEN dtx_off) oscillating with the self_good signal. The combination of these rules leads to the expert's output oscillating between turning the DTX payload on and off. Expert s_5_2 exerts more influence on the ensemble output and turns on the DTX payload. Whether these activations are correct is up to the designer to validate. However, this kind of analysis is straightforward to do and easy to understand which support manual adaptation of the system design.

Node B5 has poor stored energy performance as seen in Figure 9.13. This is due in part to the over-zealous activation of the DTX payload. However, sensor 3 is also active constantly which draws significant power. Stored energy does not reach zero, however better rule design or fuzzy set parameterization may lead to a higher average energy level while maintaining sensor activity.

The scoring applied to this example problem gives points for having stored energy greater than 25 Wh, being in the 'self good' state, and activating sensors and the high-bandwidth transmitter. Bonus scoring is achieved when multiple sensors or the transmitter are on at the same time. A score multiplier is applied to payload activation scoring when the target is in view. Figure 9.14 shows the total score (including energy and self-health) and Figure 9.15 shows subtotal score (payloads only).

Table 9.1 Sensor nodes performance comparison

Hierarchy	Total	Subtotal	Self Good	Stored Energy
Grand Total	5306937	2670762	1838960	797215
Grand Total (avg)	28686	14436	9940	4309
SEN_A	1562072	605287	626045	330740
SEN_A (avg)	31241	12105	12520	6614
SEN_A (max)	41285	15982	14400	14400
SEN_A (min >0)	14923	973	10	215
SEN_B	630010	268190	281715	80105
SEN_B (avg)	21000	8939	9390	2670
SEN_B (max)	29880	12404	14400	14400
SEN_B (min >0)	12845	370	15	50
SEN_C	950357	517387	284565	148405
SEN_C (avg)	31678	17246	9485	4946
SEN_C (max)	50454	21674	14400	14400
SEN_C (min >0)	25734	2377	4970	90
SEN_D	2164498	1279898	646635	237965
SEN_D (avg)	28859	17065	8621	3172
SEN_D (max)	51536	22756	14400	14400
SEN_D (min >0)	24819	20124	4695	55
A6	29703	1033	14270	14400
B5	16494	10884	4760	850
C28	31519	2719	14400	14400
D34	31869	20674	11080	115

Compared to the B orbit average, node B5 does worse in total score based on its small amount of time spent in the 'self good' state and above the battery threshold. However, B5 outscores its average peer in payload activation.

In general, node B5 can be considered successful. It never reaches a zero-power state. It takes observation data when the target is visible. It activates sensor 3 opportunistically. It has done all of this with a general ruleset and parameterization common across all sensor nodes.

9.2.2 Node D34 - Type 2 - Nominal

Node D34 is part of the type 2 sensor group. It has an optical camera and two range payloads (radar and lidar) and generates solar power at 42 Watts. The D orbit is at a high altitude (725 km) Sun-synchronous orbit (SSO) with inclination (98°). Orbit D satellites have frequent contact with communications nodes and occasional links to processor

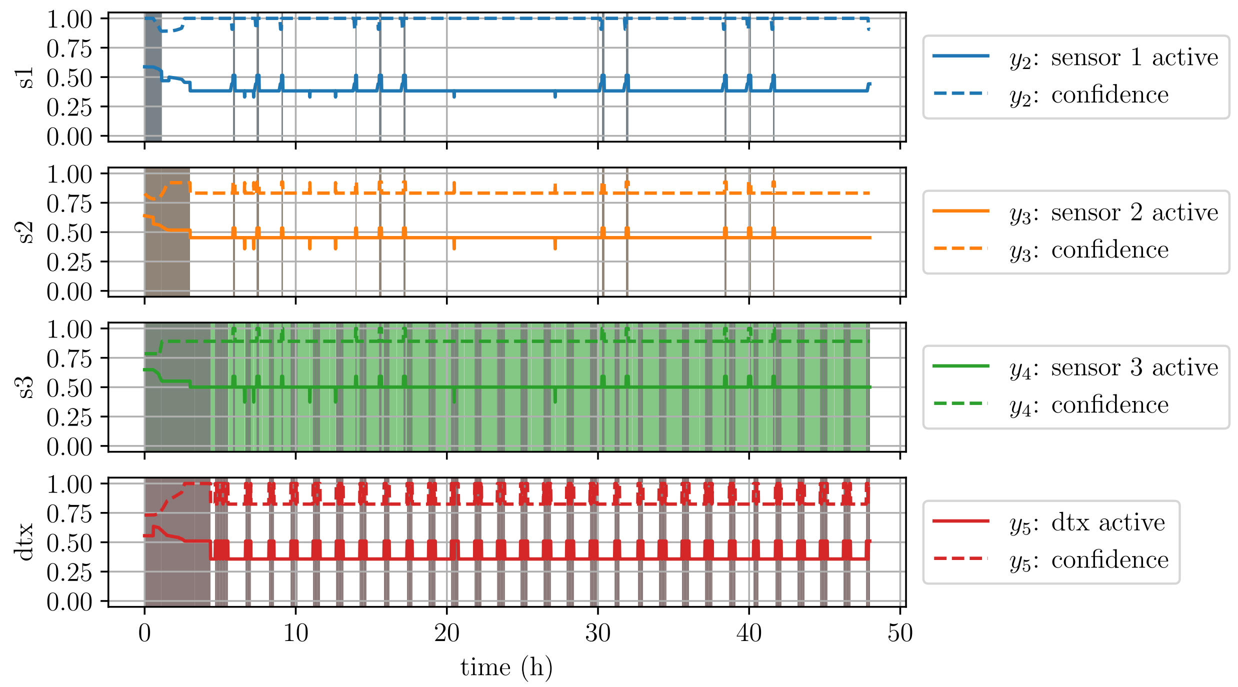


Figure 9.8 Node B5: payload activation and scoring

nodes. The Sun-synchronous orbit provides good visibility to the ground target on a regular schedule at approximately the same time each day.

Figure 9.16 shows when each sensor and high-bandwidth communications payload are active and scoring points for activity. Similar to node B5, the spikes in sensor activation correspond to ground target visibility shown in Figure 9.17. This is good behavior.

Sensors 2 and 3 activate together regularly providing many high-scoring opportunistic observations. The data transmission also activates often which is appropriate given the frequent communication node links available. Figure 9.18 shows when communication and processor nodes are in range.

Stored energy performance for D34, shown in Figure 9.19, is similar to node B5. Node D34 has a higher average level and more frequent critically-low dips that correspond to the target being visible while the satellite is not sunlit. Still, the satellite does not drain its battery to zero.

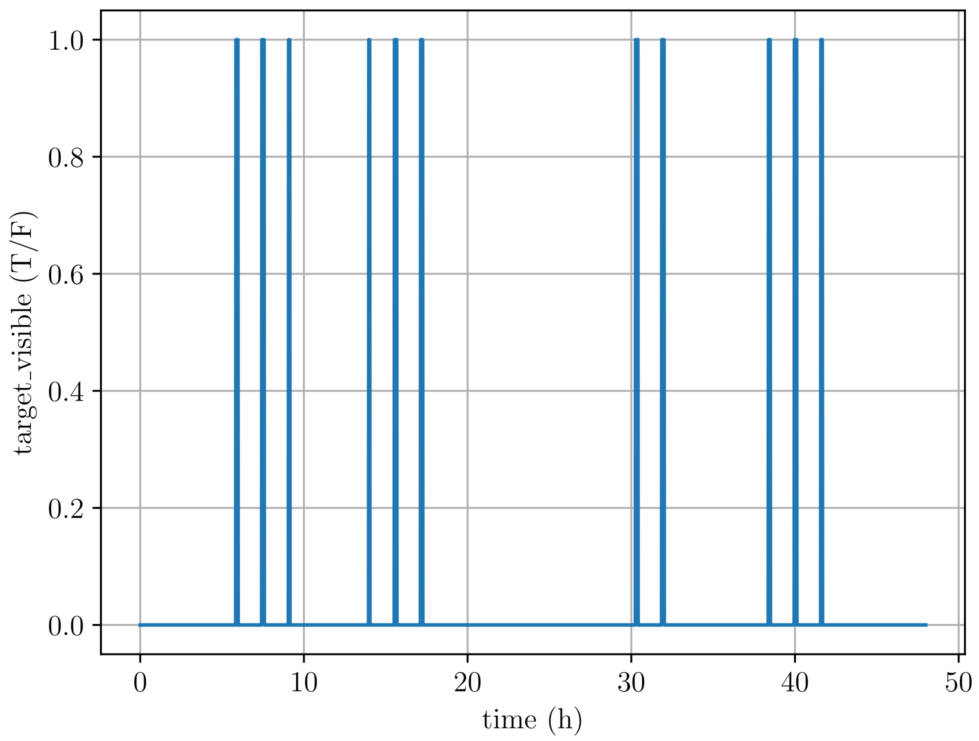


Figure 9.9 Node B5: ground target visibility

Node D34 outperforms its peers in all but battery status scoring. Compared to node B5, D34 attains almost twice the score. This is in part due to the higher score values assigned to its sensors but can also be attributed to more frequent multi-payload activation. Figure 9.20 shows the total score (including energy and self-health) and Figure 9.21 shows subtotal score (payloads only).

Node D34 is an exemplary performer in this network simulation. It outperforms the average score across all sensor orbits. It maintains nonzero stored energy in spite of multiple occurrences when the target is in view in the dark.

9.2.3 Node A6 - Type 1 - Faulty

Node A6 is another type 1 satellite in a moderate altitude (550 km) and high inclination (98°) Sun-synchronous orbit. However, this node is faulty. The simulated fault manifests as four times the normal power draw and exists for 99.9% of the time.

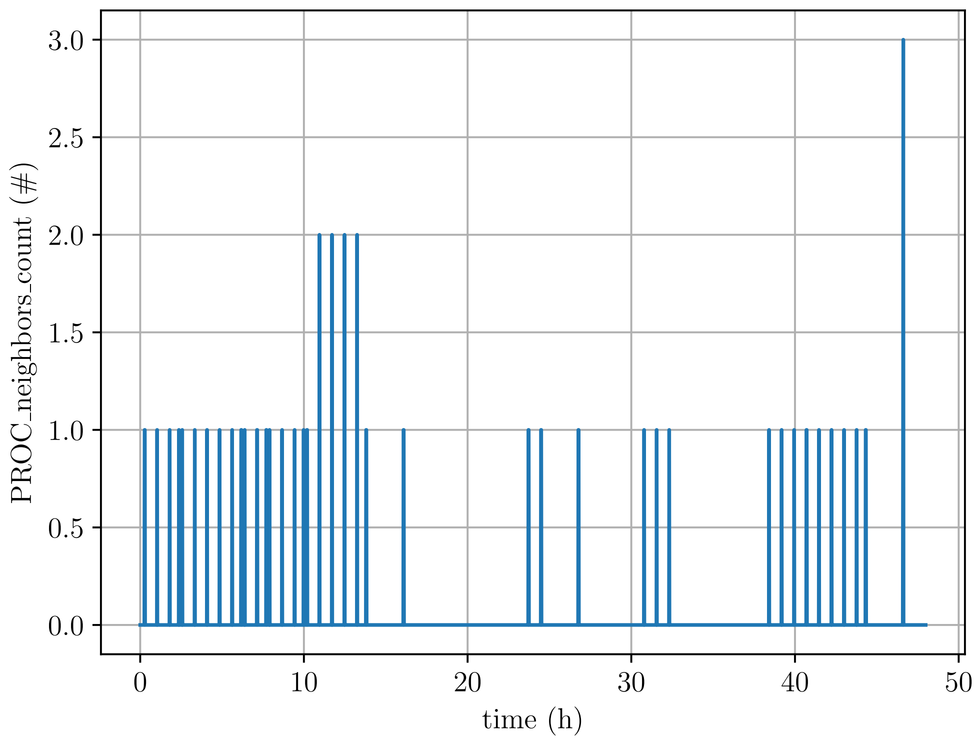


Figure 9.10 Node B5: processor nodes in range

As shown in Figure 9.22 the fault condition all but eliminates scoring opportunities for this node. The sensor payload activation scoring occurs when the target is in view (see Figure 9.23). Turning sensors on while the satellite is in a known-faulty condition could be risky. However, the added value of observing the target offsets these risks and is therefore considered excellent behavior from this sensor node.

Figure 9.24 illustrates how stored energy benefits from the fault condition enforcing sensor payload deactivation. This puts the satellite in a power-positive state ensuring ample resources for fault mitigation strategies. The power-positive state also meets the stored energy scoring threshold which drives about half of the node's total score as shown in Figure 9.25. The spikes in score rate correlate to when the sensors are active which contributes about 3% of the total score. The other half of the total score is primarily due to activation of the 'self_good' output signal. This could be considered erroneous as there is a known fault. However, assessing status as good given the fault is a useful metric and is mostly driven by

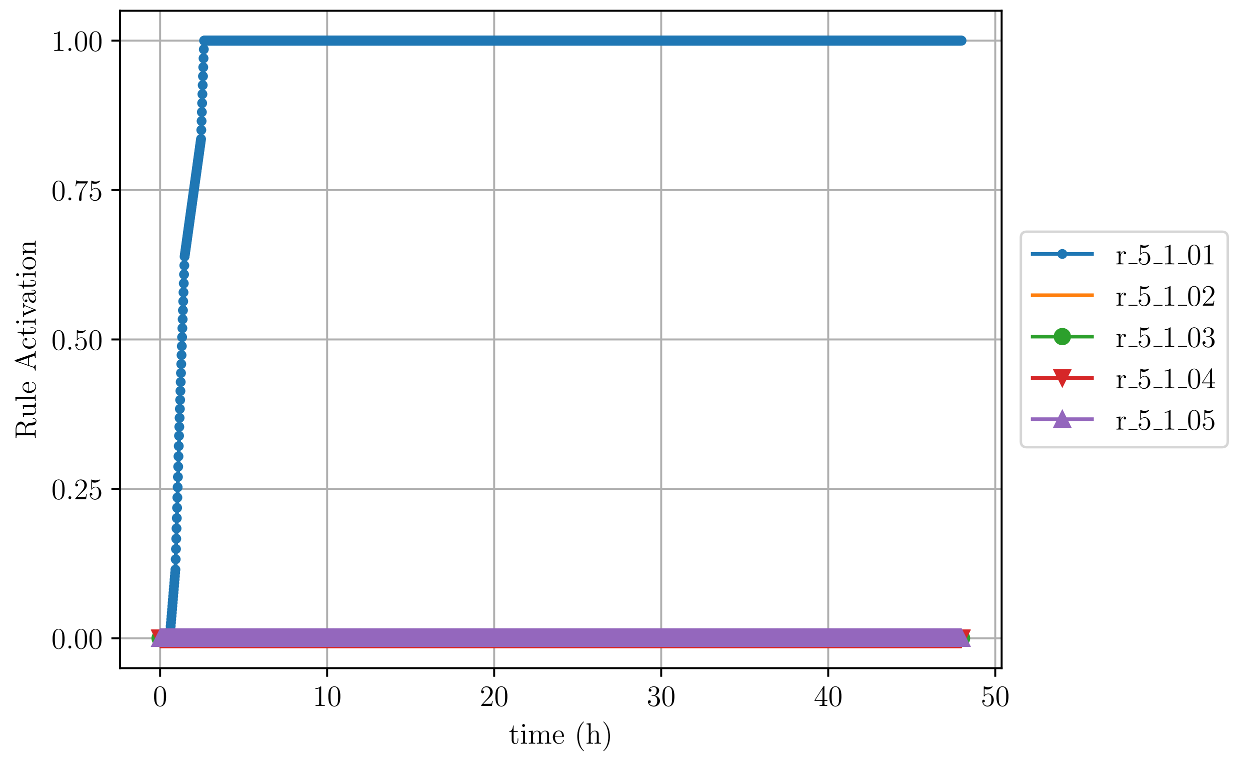


Figure 9.11 Node B5: expert s_5_1 rule activation

the stored energy state (see Figure 9.26 which shows that self_good is highly sensitive to the stored energy state).

Overall, node A6 shows expected performance given its faulty state. The activation of the sensors when the target is in view is a net-positive driven by the ruleset. Stored energy is maximized to support fault mitigation. The system is rewarded for maintaining the battery status through both battery and self_good threshold scoring.

9.2.4 Node C28 - Type 2 - Faulty

Similar to node B5, node C28 is in a low (350 km) mid-inclination (45°) orbit. This satellite is a type 2 vehicle with the higher-power sensor and solar generation systems. It is also a faulty node with the same quadrupled power draw as node A6.

The node's performance (shown in Figures 9.27-9.28) is similar to node A6 in that the scoring is driven by the stored energy and self_good threshold scoring with some sensor activation when the target is in view.

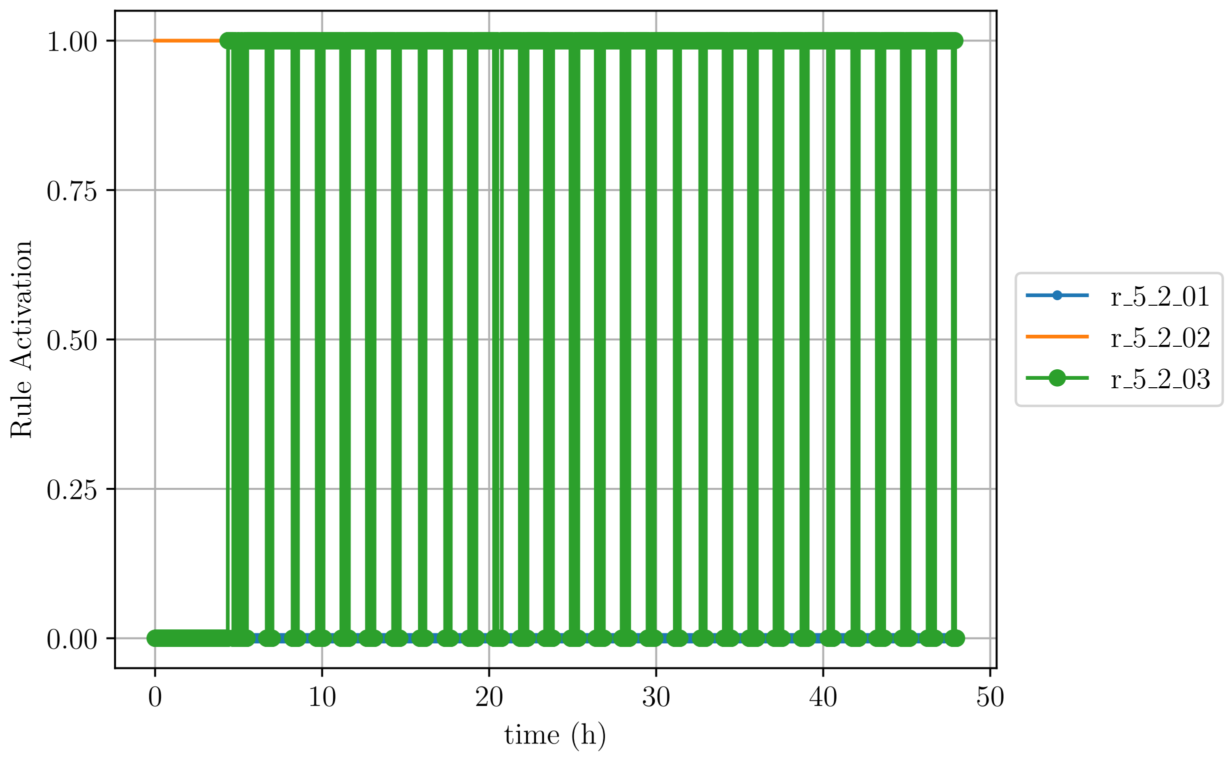


Figure 9.12 Node B5: expert s_5_2 rule activation

9.3 Baseline Performance Conclusions

In general, the sensor network performed well given the initial rule and fuzzy set parameters design. All sensor nodes activated their payloads when the ground target was in view. The faulty nodes maintained stored energy levels and also turned sensors on driven by high-value observations of the target.

However, there were a few notable and suboptimal behaviors. First is the activation of the high-bandwidth data transmitter regardless of whether a communications or processing satellite node is in range. There were satellites that were power-negative enough to fully drain their batteries which results in degraded mission capability. These issues are a result of suboptimal rule definitions and fuzzy set parameters driving rule activation. The mitigation is to tune and adapt the system based on the introspection metrics and the simulated performance indicators. A manual tuning process is described and implemented in Chapter 10.

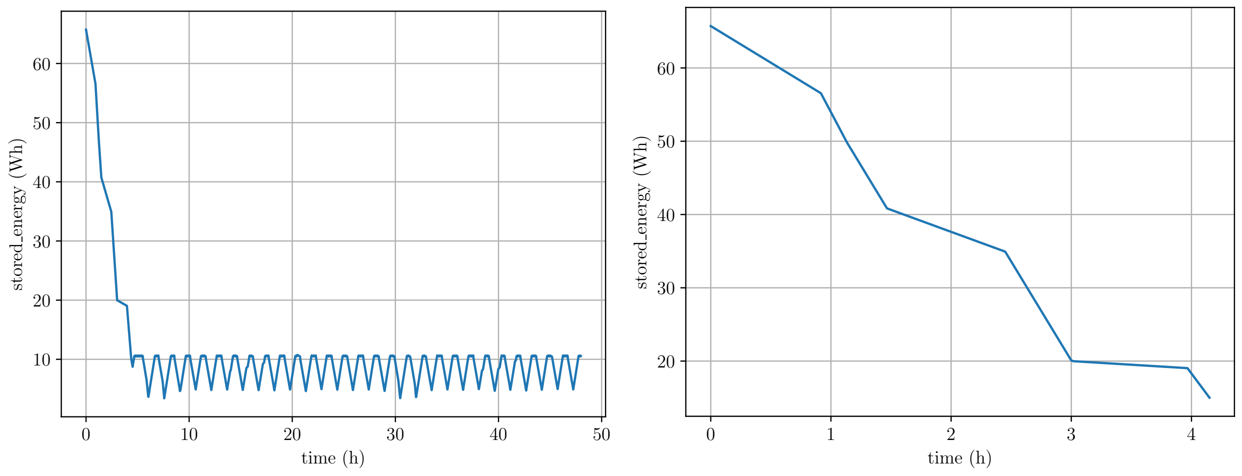


Figure 9.13 Node B5: stored energy state

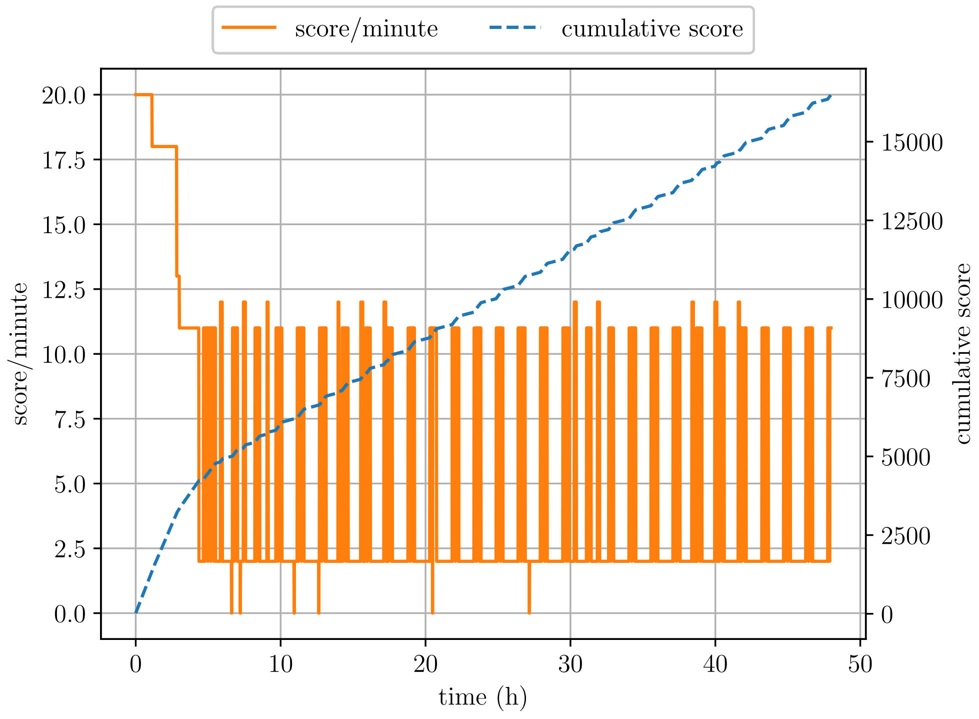


Figure 9.14 Node B5: total score over time (including battery, self good, and payload scoring)

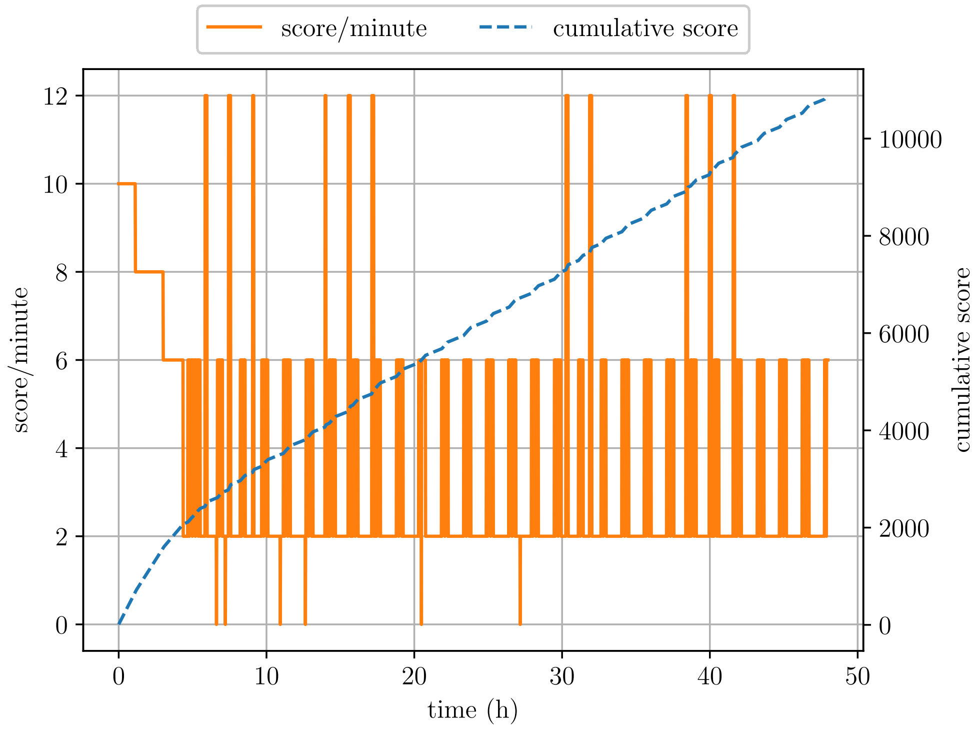


Figure 9.15 Node B5: subtotal score over time (including payload scoring)

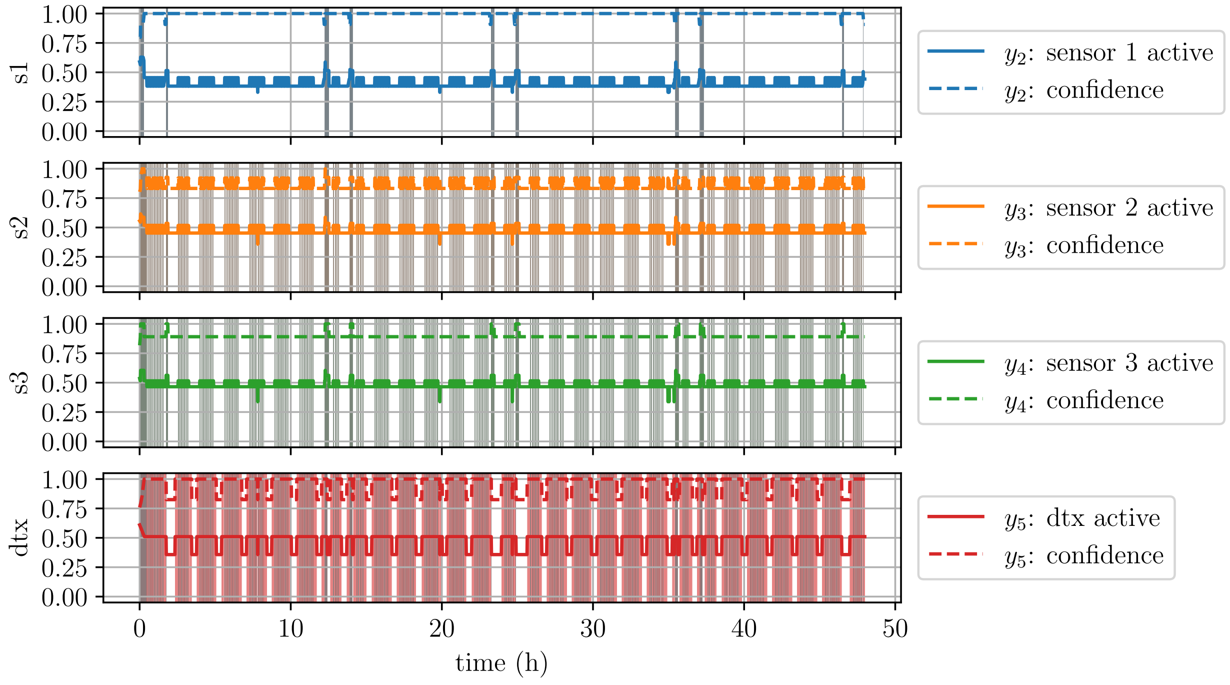


Figure 9.16 Node D34: payload activation and scoring

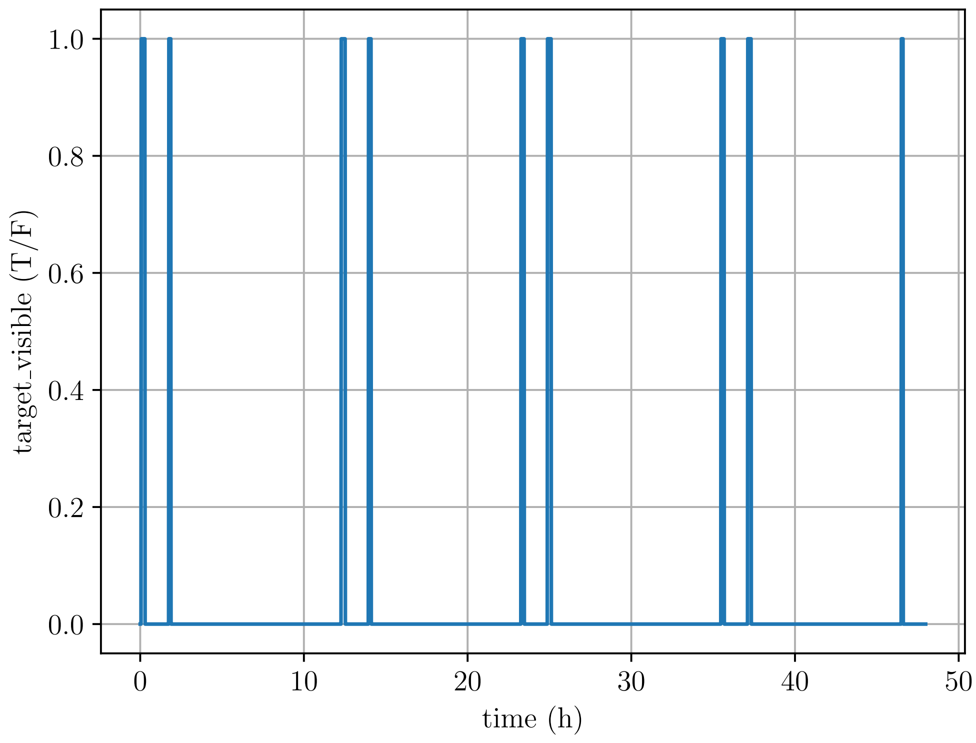


Figure 9.17 Node D34: ground target visibility

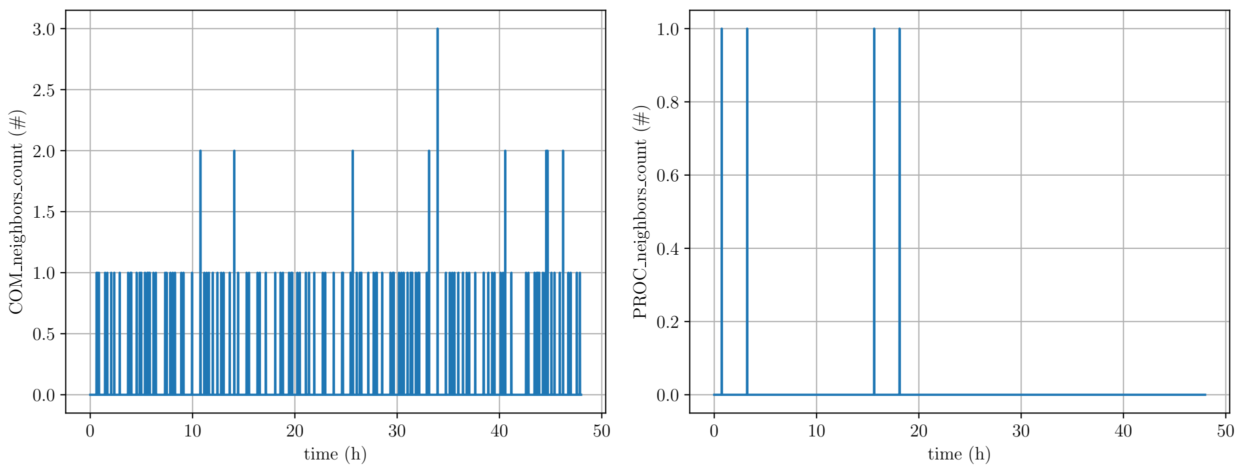


Figure 9.18 Node D34: communication and processor nodes in range

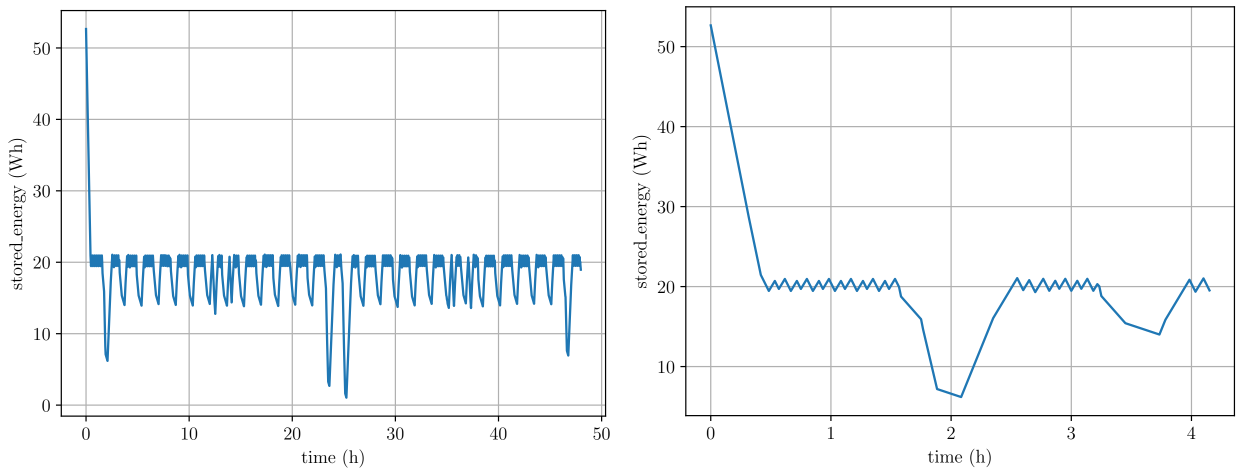


Figure 9.19 Node D34: stored energy state

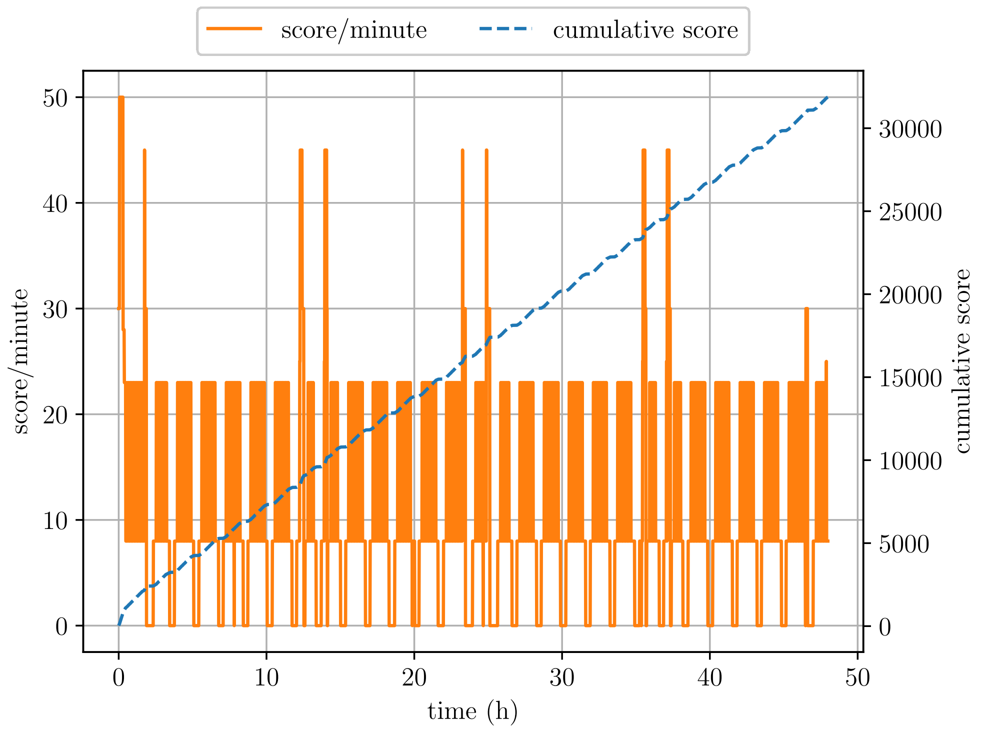


Figure 9.20 Node D34: total score over time (including battery, self good, and payload scoring)

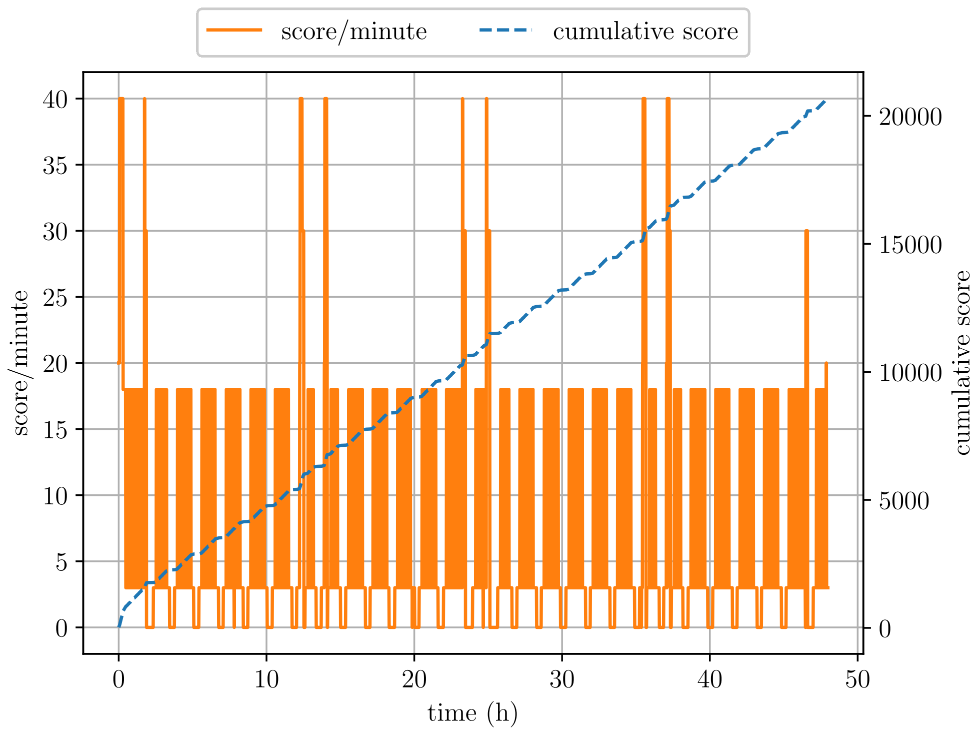


Figure 9.21 Node D34: subtotal score over time (including payload scoring)

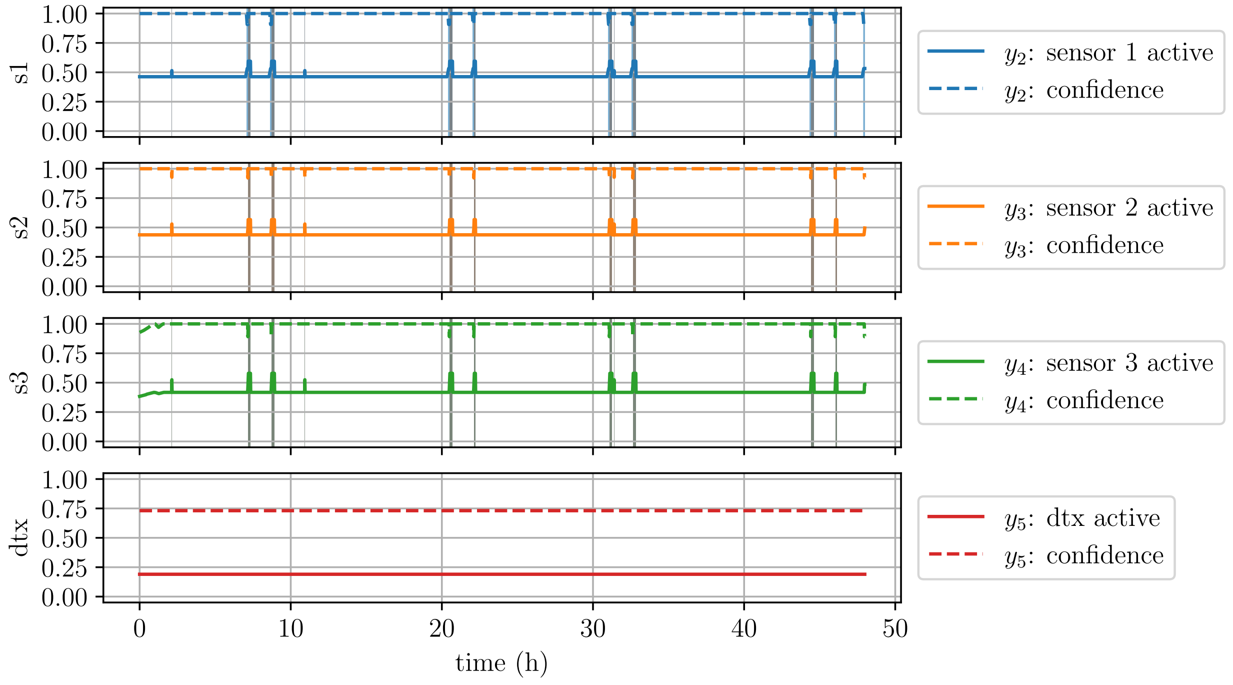


Figure 9.22 Node A6: payload activation and scoring

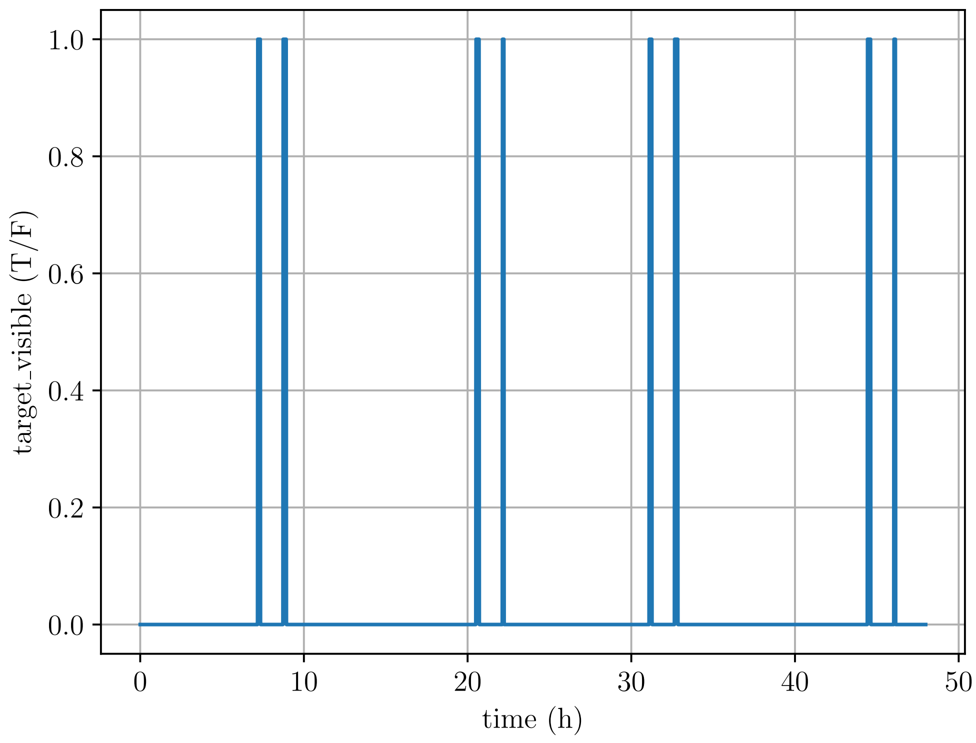


Figure 9.23 Node A6: ground target visibility

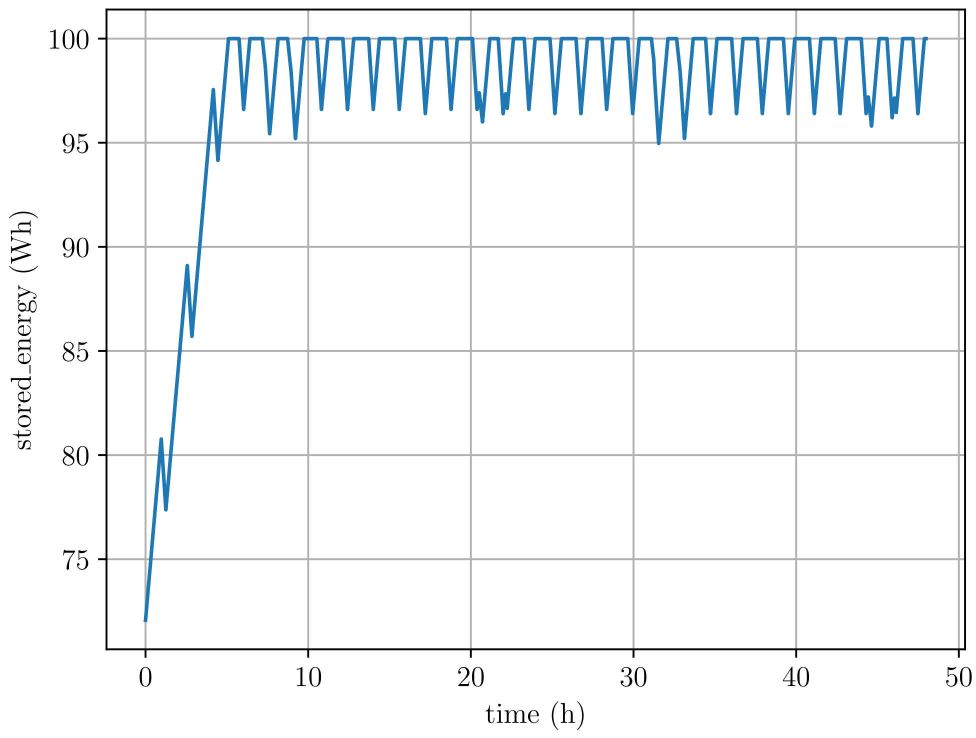
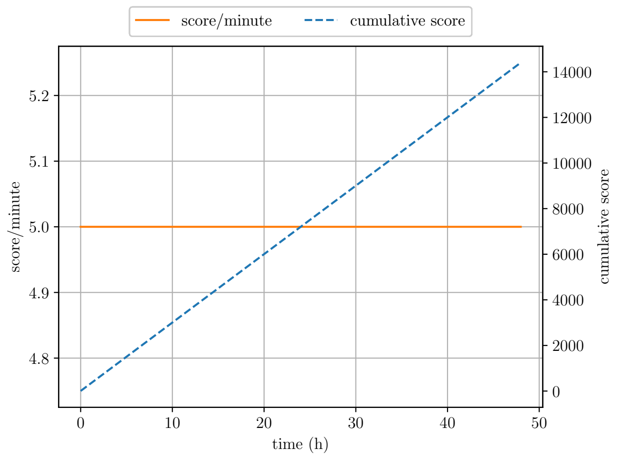
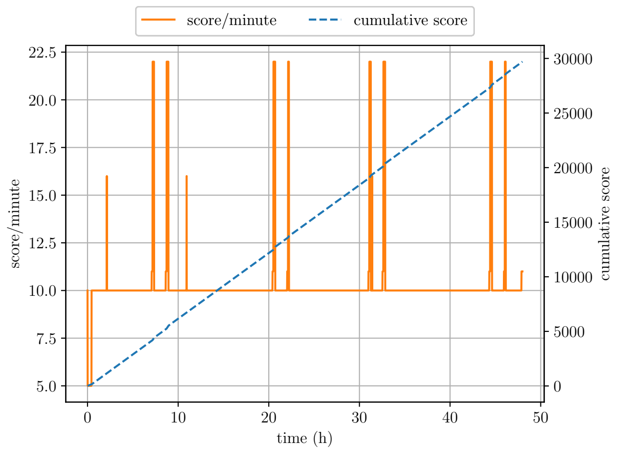


Figure 9.24 Node A6: stored energy



(a) Stored energy score



(b) Total score

Figure 9.25 Node A6: stored energy score and total score

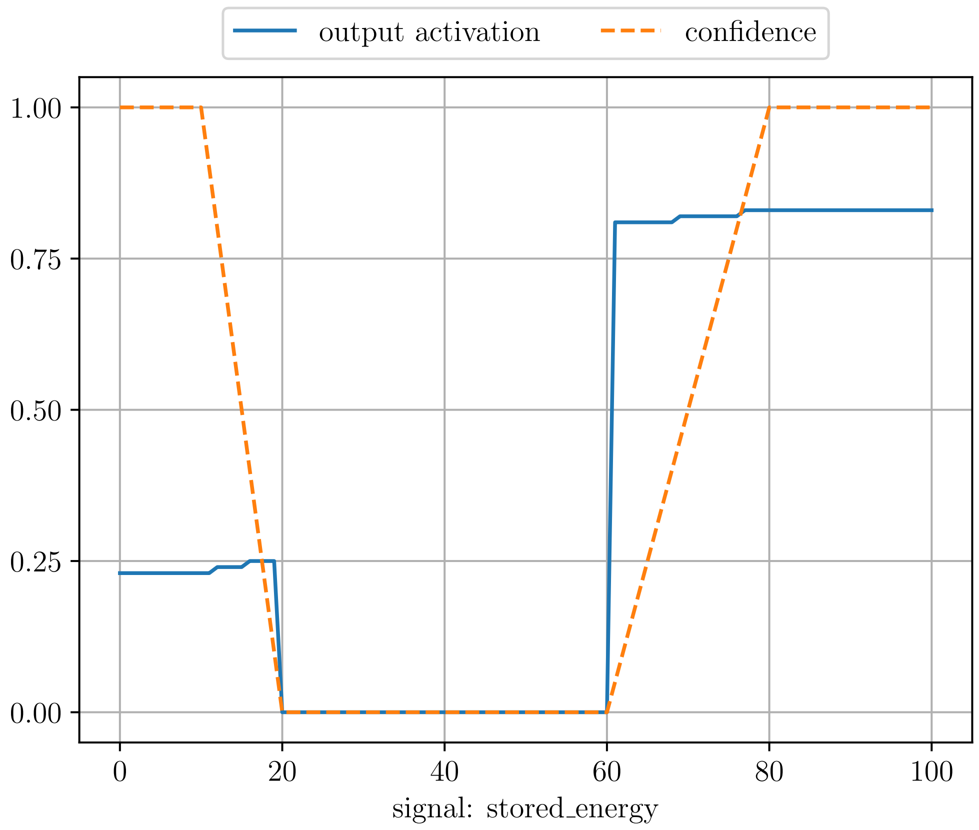


Figure 9.26 Node A6: self_good sensitivity to stored energy status

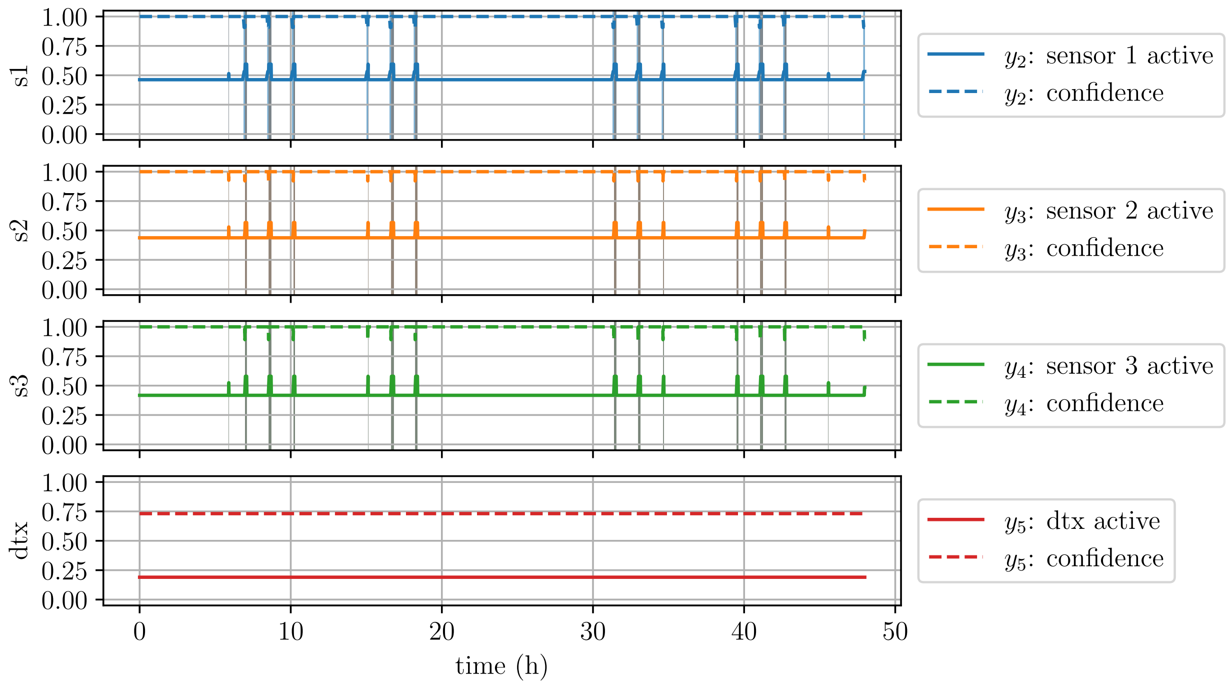


Figure 9.27 Node C28: payload activation and scoring

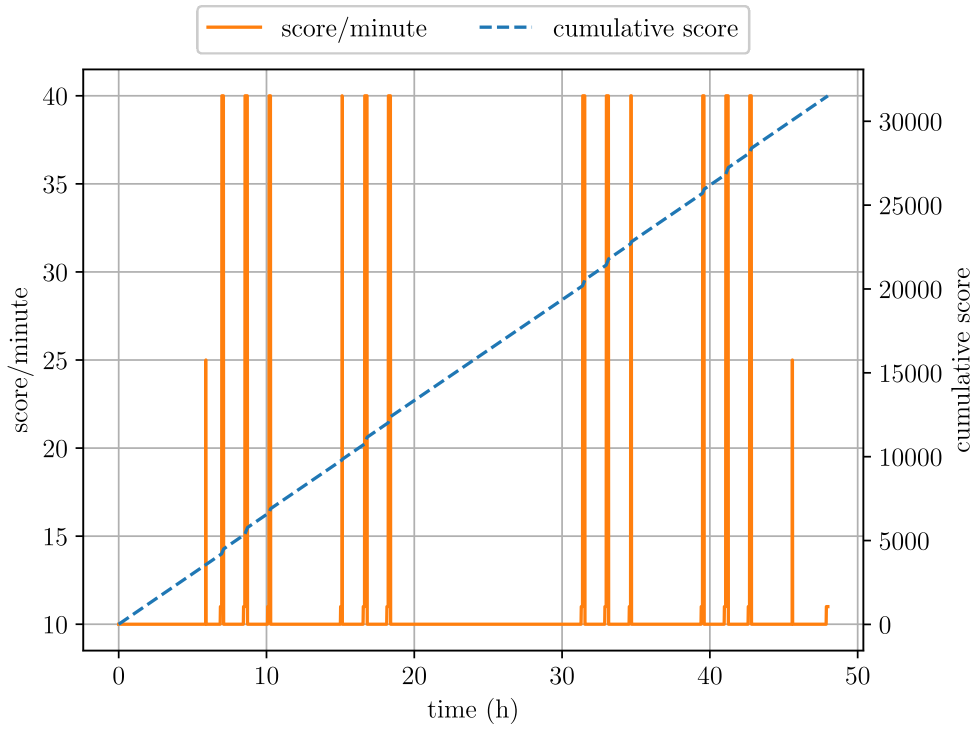


Figure 9.28 Node C28: total score over time (including battery, self good, and payload scoring)

Chapter 10: Performance Optimization Approaches and Results

The introspective tools presented in Chapter 6 are useful for understanding system complexity and optimizing the ruleset to avoid redundancy. Sensitivity analysis can also inform a designer which ensembles, experts, or rules to further scrutinize. Outputs that do not show expected sensitivity to variations in inputs will lead the designer toward high-return modifications.

In running the HEADS system through simulation the system designer can also capture and analyze individual fuzzy sets and rules. This level of insight is possible because of the fuzzy linguistic variables in use. Considering which rules activate based on which fuzzy sets activate, and when, can help the designer find and fix suboptimal rules and fuzzy sets.

The rest of this chapter is in two parts. The manual tuning changes are presented with explanations, and then the performance results are presented with comparison to the baseline system.

10.1 Manual Tuning Approach

A manually-adapted system was developed using the baseline system and analysis of the simulation results. Careful review of the sensitivity data, fuzzy set activation, and rule activation over time led to the following conclusions:

1. the baseline system was too quick to activate the 'self_good' output when power states were low,

2. the high-bandwidth transmitter was activated too often, and

3. the sensor activation favored sensor 3 when the target was not in view.

These assessments drove the manual fuzzy set parameter tuning adaptations that are shown in Table 10.1. Two rules were also changed:

1. Ensemble 5 (dtx_active), Expert 1 (neighbor focus), Rule 2:
    - Baseline: batt_bad AND ((dt_sun_short OR sunlit) AND (COM_neighbors OR PROC_neighbors)) THEN dtx_on
    - Adapted: *batt_good* AND ((dt_sun_short OR sunlit) AND (COM_neighbors OR PROC_neighbors)) THEN dtx_on
2. Ensemble 5, Expert 2 (fault focus), Rule 2:
    - Baseline: NOT faulty AND NOT fault_recovery THEN dtx_on
    - Adapted: NOT faulty AND NOT fault_recovery *AND self_good* THEN dtx_on

The first rule change is intended to make the 'dtx_active' output less likely when stored energy is low. The second rule change encourages waiting until the 'self_good' output is active before activating the high-bandwidth transmitter.

Table 10.1 Fuzzy set parameter updates from manual tuning process

y_j,	$s_{j,p}$	Antecedent/ Consequent	Fuzzy Set	Activ. Func.	Baseline	Tuned
y_1	$s_{1,1}$	antecedent	batt_critical	ltrap	[10, 20]	[20, 30]
		antecedent	batt_low	trap	[15, 20, 25, 30]	[15, 25, 35, 60]
		antecedent	batt_good	trap	[30, 50, 70, 80]	[50, 60, 70, 90]
		antecedent	batt_full	rtrap	[60, 80]	[80, 90]
		consequent	self_bad	ltrap	[0.4, 0.5]	[0.4, 0.45]
		consequent	self_marginal	trap	[0.45, 0.5, 0.55, 0.65]	[0.4, 0.5, 0.55, 0.65]
y_3	$s_{3,3}$	consequent	sensor_on	rtrap	[0.35, 0.45]	[0.35, 0.95]
	$s_{3,4}$	consequent	sensor_on	rtrap	[0.35, 0.7]	[0.35, 1.7]
y_4	$s_{4,1}$	consequent	sensor_off	ltrap	[0.5, 0.6]	[0.3, 0.6]
		consequent	sensor_on	rtrap	[0.5, 0.6]	[0.3, 0.6]
	$s_{4,2}$	consequent	sensor_off	ltrap	[0.3, 0.4]	[0.3, 0.55]
		consequent	sensor_on	rtrap	[0.6, 0.7]	[0.45, 0.7]
	$s_{4,3}$	consequent	sensor_off	ltrap	[0.3, 0.4]	[0.4, 0.5]
		consequent	sensor_on	rtrap	[0.35, 0.45]	[0.45, 1.55]
	$s_{4,4}$	consequent	sensor_off	ltrap	[0.3, 0.4]	[0.4, 0.5]
y_5	$s_{5,1}$	antecedent	batt_bad	ltrap	[30, 60]	[40, 50]
		antecedent	batt_good	rtrap	[30, 30]	[40, 80]

10.2 Manual Tuning Results

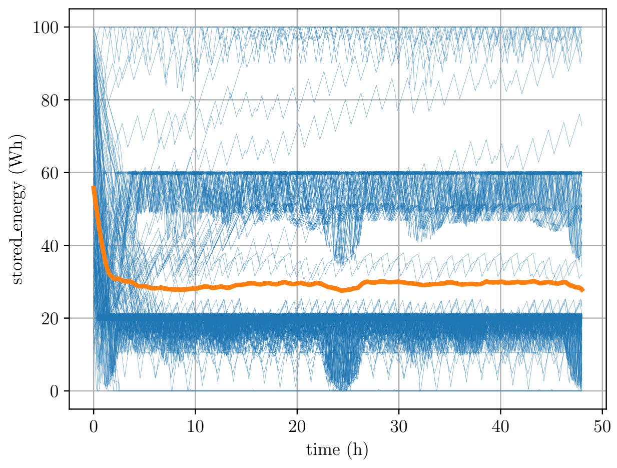
The simulation was re-run using the modified rules and fuzzy set parameters. Compared to the baseline system the tuned system as a whole saw performance increase on average by:

- 48% in total score,
- 17% in sensor activation scoring,
- 34% in self_good scoring, and
- 185% in stored_energy scoring.

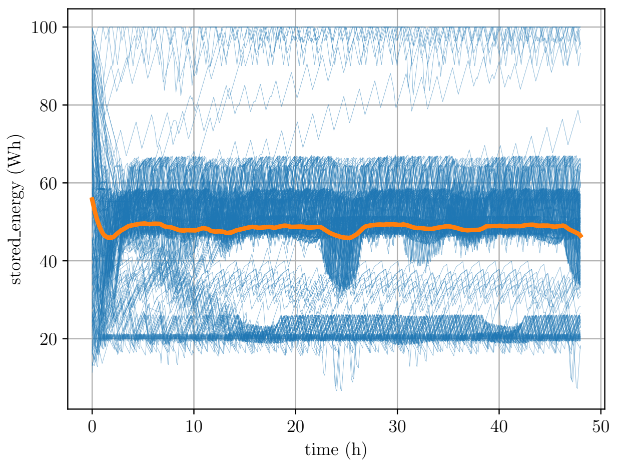
Complete scoring comparisons at the system, orbit, and per-satellite level are shown in Table C.1 in Appendix C.

The increase in performance is visible across the system states and outputs. Figure 10.1 shows the stored_energy state for all sensor nodes for the baseline and manual tuning simulations. There are no satellites that had stored energy go to zero at any time during the tuned simulation. This is compared to 18 of 185 satellites with the baseline system design that had some time at zero energy.

Figure 10.2 shows the stored_energy score for sensor node D34 compared between the baseline and tuned simulations. The average stored energy for this node remains much higher after system tuning leading to score being awarded for the duration of the simulation. The increased minimum power does not come at the cost of sensor activation with the tuned system resulting in a score that is 8% higher than the baseline system as seen in Figure 10.3. Finally, Figure 10.4 shows the sensor activation and scoring for node D34. Compared to the baseline system (see Figure 9.16) the tuned system activates sensors 1 and 2 more frequently and almost always together. Sensor 3 still sees duty when the ground target is in view. This is desirable as it maximized opportunistic sensing while maintaining full sensor activity when the desired target is visible. This proves that the rule and fuzzy set parameter



(a) Baseline



(b) Tuned

Figure 10.1 Stored energy state comparison between the baseline and tuned systems

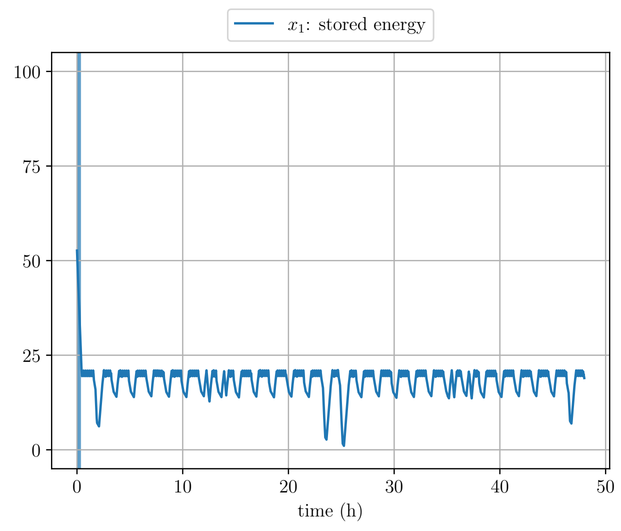
modifications successfully reduced the preference for sensor 3. However, the high-bandwidth transmitter sees constant activation likely due to the increased stored energy state. This is perhaps a good case for structural modifications to the ensemble through the addition of one or more specialized experts to enforce neighbor visibility constraints.

10.3 Evolutionary Optimization Approach

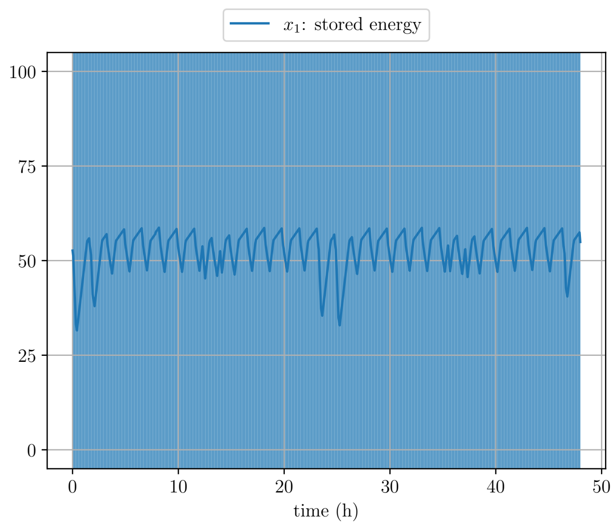
Evolutionary optimization through a modified genetic algorithm using real-valued chromosomes is applied to the HEADS system designed for this example problem. The manually-tuned parameter set was used as the new baseline for genetic tuning. All results are presented as compared to the manually-tuned results. The evolutionary optimization approach is discussed in more detail in Chapter 7. This section provides an overview of how the genetic algorithm was applied to the example HEADS system.

10.3.1 Evolutionary Mechanisms

The structure of the HEADS ensembles, experts, and fuzzy sets makes up the system genotype that remains constant through evolution. A phenotype is a specific set of parameters within the genotype structure and contains the complete set of fuzzy set parameters describing a population member.



(a) Baseline



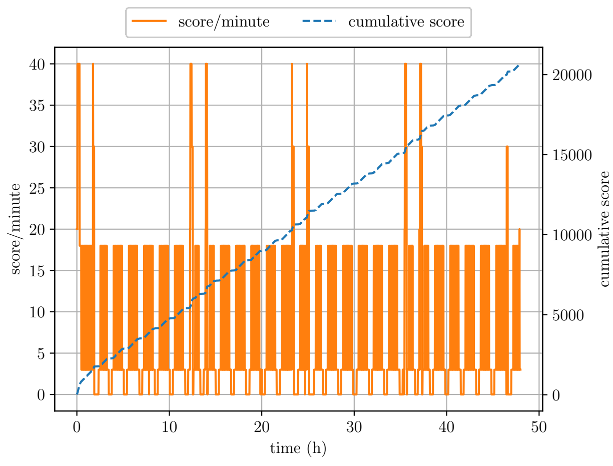
(b) Tuned

Figure 10.2 Node D34 stored energy scoring comparison between the baseline and tuned systems

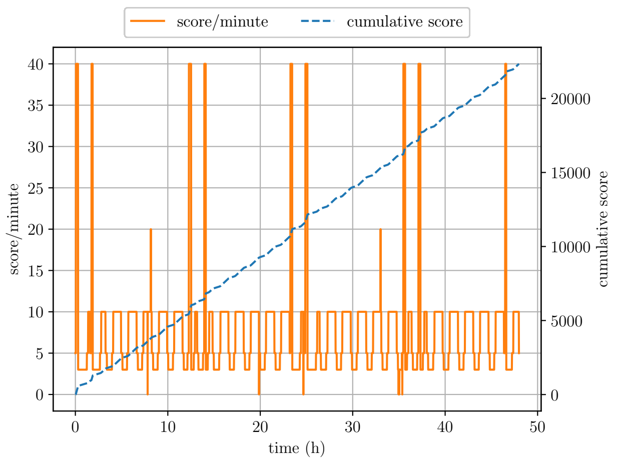
Evolutionary optimization takes the genotype (structure) and generates new phenotypes (parameter sets) to test. Three main evolutionary methods are employed to adapt the system:

1. local stochastic expansion,
2. child phenotype generation from two parent phenotypes, and
3. mutated phenotype creation from one source phenotype.

Local stochastic expansion is a localized search method that creates new phenotypes that are normally distributed around the original phenotype. The goal for local search is to find a slightly different parameter set that may prove to be more performant that the previous local optimum. Generating child phenotypes using two parent phenotypes is a stochastic combination of about half of the parameters from one parent with the complementary half from the other parent. The new phenotypes have a chance to perform better than either parent if both parents contribute component parameters that performed well over different



(a) Baseline



(b) Tuned

Figure 10.3 Node D34 sensor activation scoring comparison between the baseline and tuned systems

areas of the input space. Finally, mutation is a way to explore beyond local optima by high-variance stochastic modification of a small number of parameters.

10.3.1.1 Constrained Evolution

There are multiple ways to implement and constrain the genetic algorithm modifications including setting the variance for stochastic methods and allowing or preventing out-of-bounds parameters. Because of how the fuzzy set parameter activation functions are defined, the HEADS system enforces monotonically-increasing parameters for each fuzzy set. If a random parameter is created that is out of order with the others in its set, the new parameter is used as an anchor and the other parameters changed to be in order.

Genetic mutation was also constrained to enforce fuzzy set parameters to remain within pre-defined minimum and maximum signal bounds. The designer sets minimum and maximum logical signal values on each input and output signal. The constraints are enforced in two ways, including:

- ensuring the mutated parameter is within the signal minimum and maximum value, and

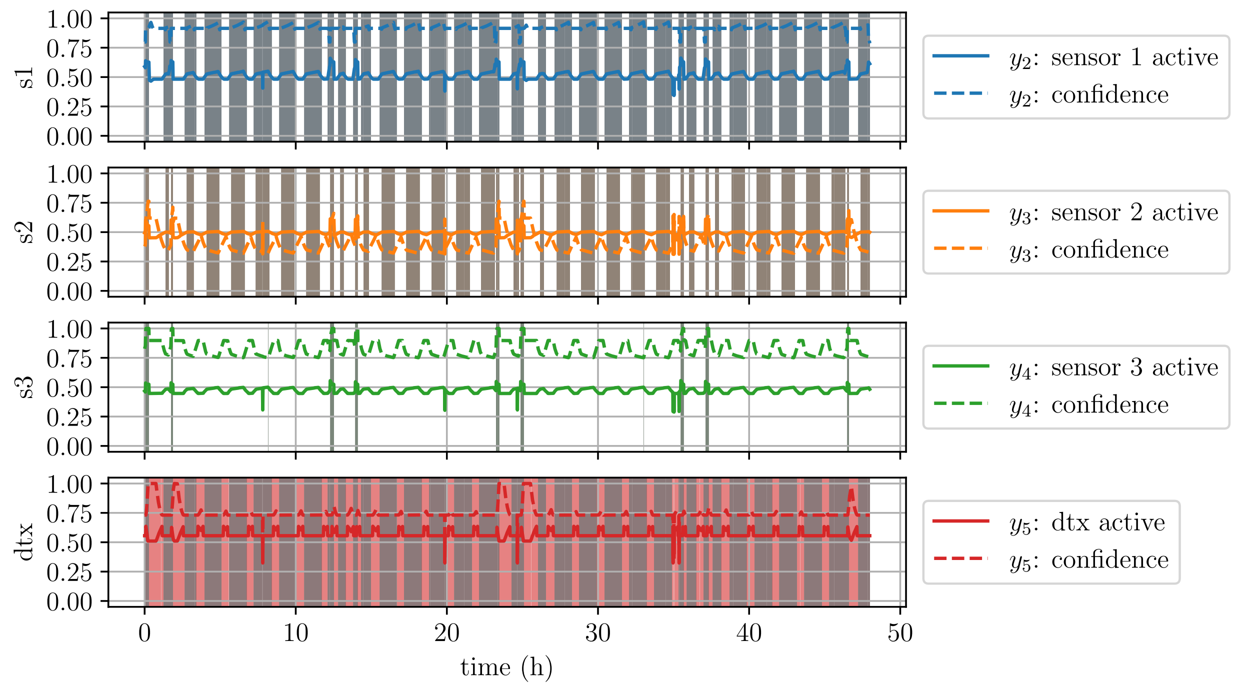


Figure 10.4 Node D34: payload activation and scoring for the tuned system

- forcing the other parameters for the fuzzy set to adapt around the mutated parameter while remaining within the signal bounds.

If a mutated parameter is less than a pre-defined lower boundary value or greater than the upper boundary, it is re-created by random sample between the boundary values. For example, suppose the signal is defined between $[0, 10]$ and the second parameter mutates to -7 with its mutated set being $[2, -7, 6, 12]$:

- the invalid mutated parameter (-7) is re-sampled from a uniform distribution between the valid boundaries yielding, for example, $[0, 10] = 1.5$, and

- the other parameters are adapted for monotonicity and boundary compliance, e.g., $[1.35, 1.5, 6, 9.1]$.

10.3.1.2 Logical Ordering Constraints

The designer could also enforce inter-fuzzy set logical ordering where 'batt_low' must always be defined on a lower part of the signal domain than 'batt_full'. This constraint was not used for this example problem because allowing unconstrained set domain ordering can provide insight into expert and rule structure issues.

Figure 10.5 shows an example of this where the ‘batt_good’ fuzzy set is fully encompassed within the ‘batt_full’ set. While this is still logically valid it does indicate that the ‘batt_good’ fuzzy set may not be used or have great impact on the expert’s output and could be a candidate for removal to reduce expert complexity.

This assessment can be further validated through sensitivity analysis (Figure 10.6) and reviewing rule activations based on the signals in question (Figure 10.7). The sensitivity indicates that the output does depend on the signal used but there is no discernible change based on where the ‘batt_good’ fuzzy set activates around signal value 80. The fuzzy set activation shows that ‘batt_good’ activates but has no impact on the rule activation. If the system performance is acceptable the fuzzy set ‘batt_good’ can be removed without negative impact.

10.4 Evolutionary Optimization Results

The genetic algorithm was initialized and a subset of non-faulty sensor nodes (A8, B5, C27, D34) was used during training to provide scoring data for performance analysis. A new parameter set was generated and applied to all four nodes. The entire simulation was run for these four nodes and their scores computed and saved. New parameter sets were generated using the highest-performing sets as inputs to the evolutionary mechanisms discussed below. This process was repeated until at least 50 new parameter sets were created. The highest-scoring parameter set was then applied to the entire sensor network and a full-scale simulation done to provide these results.

The performance gains from evolutionary tuning are modest compared to the increase seen from manual tuning. Compared to the manually-tuned system, the genetic tuning resulted in average system-level increases of:

- 6.8% in total score,

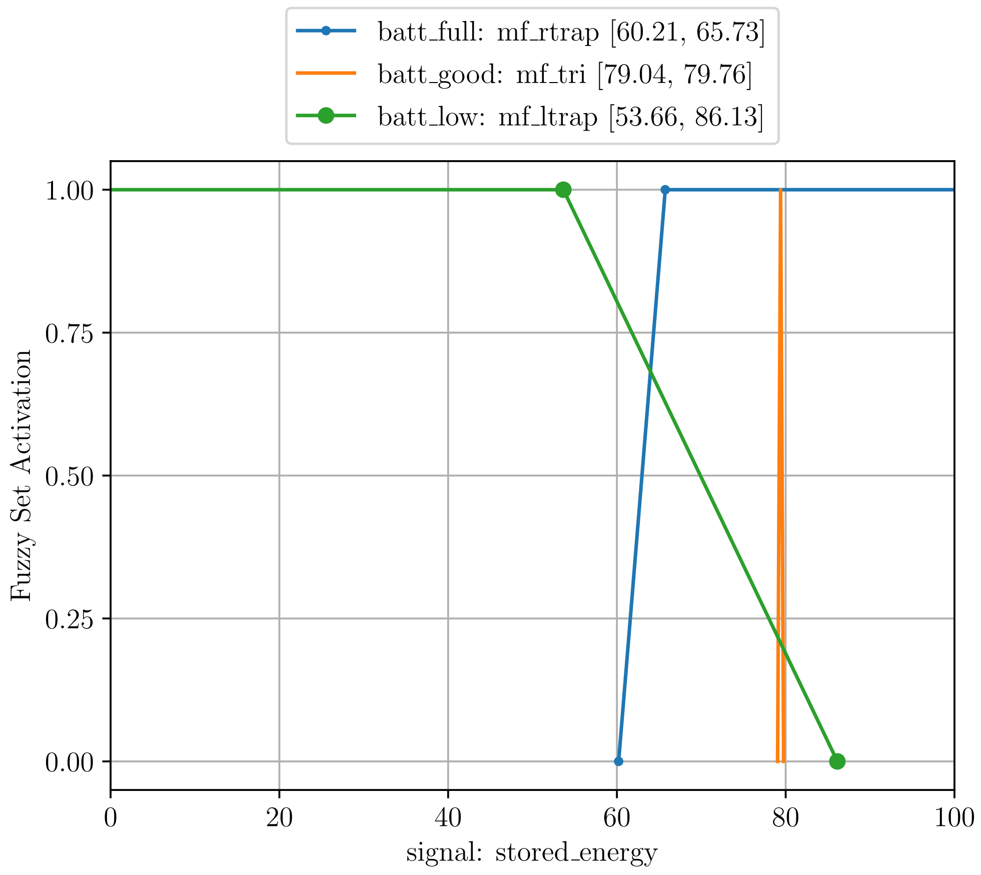


Figure 10.5 Unconstrained fuzzy set ordering for ensemble y_4 expert $s_{4,1}$

- 1.6% in sensor activation scoring,
- 4.7% in self_good scoring, and
- 16% in stored_energy scoring.

Complete scoring comparisons at the system, orbit, and per-satellite level between the baseline, manual, and genetic tuned systems are shown in Table C.1 in Appendix C.

The moderate improvements are most evident in the energy levels. Figure 10.8 shows the stored_energy state for all sensor nodes for the manual and genetic tuning simulations. Genetic tuning had one node (D29) have its stored energy go to zero briefly, compared to zero nodes for the manually tuned system and 18 for the baseline system. However, across

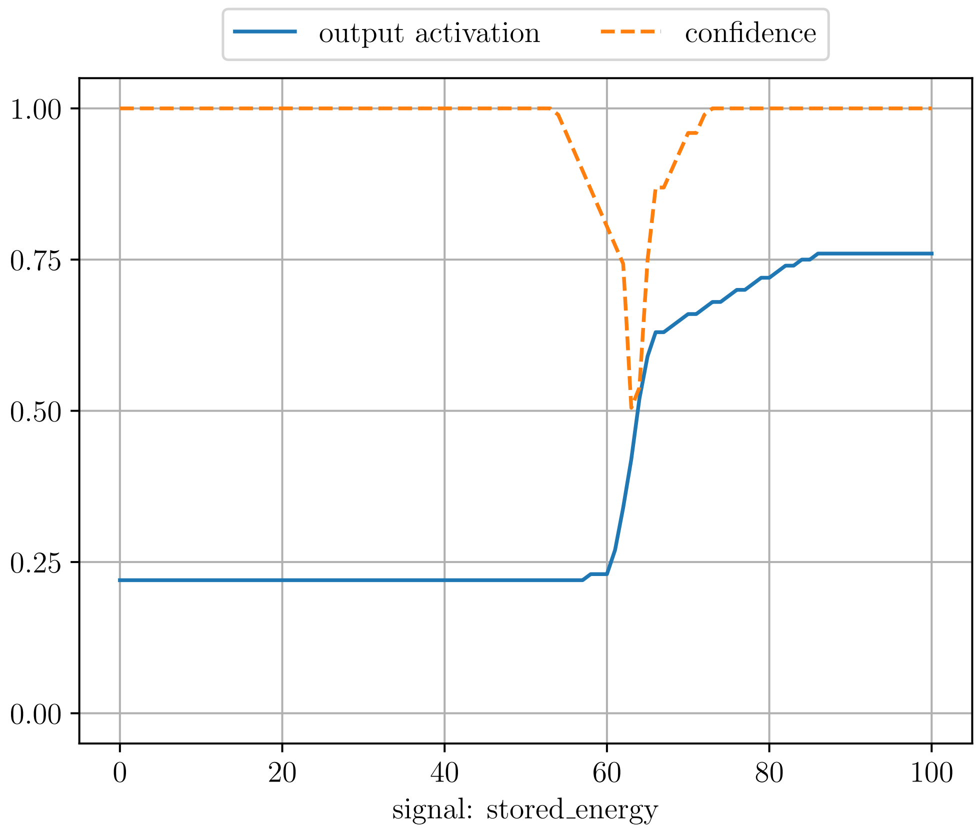


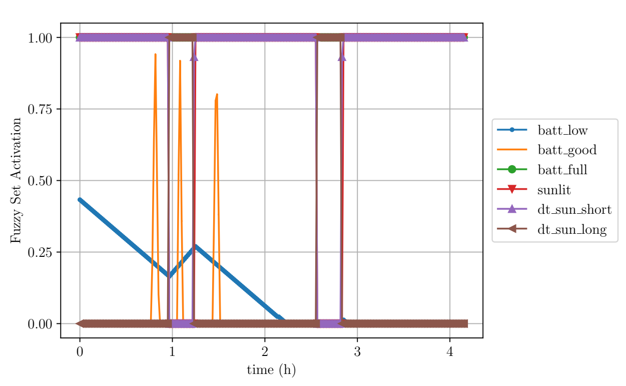
Figure 10.6 Ensemble y_4 expert $s_{4,1}$ sensitivity to input signal 'stored_energy'

the sensor network the average stored energy was slightly higher and saw fewer nodes with energy below the scoring threshold.

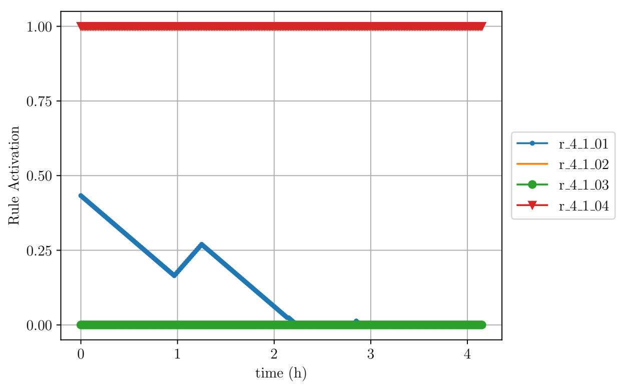
10.5 Conclusions

As shown throughout this chapter, the HEADS framework is well-suited for manual and automated tuning. Simulated performance for the example problem increased on average 48% after manual tuning and another 7% after evolutionary tuning.

The HEADS framework enables direct inspection for performance and rule activation which supports system modification. A designer or operator can review system outputs and their underlying rule and fuzzy set activations. This direct insight enables quick and meaningful conclusions about why and how the system exhibits specific behavior. It is thus straightforward to identify candidate system modifications to adapt performance and behavior.

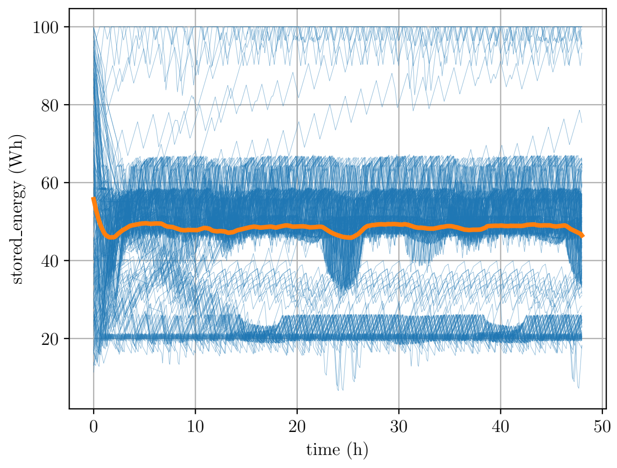


(a) **Fuzzy Set Activation**

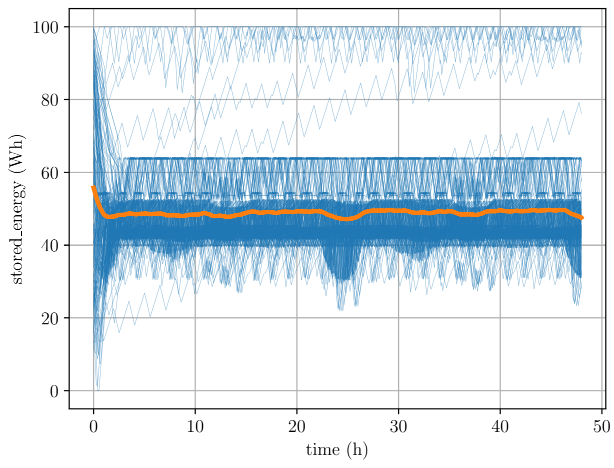


(b) **Rule Activation**

Figure 10.7 Activation of fuzzy set 'batt_good' has no impact on rule activation for ensemble y_4 expert $s_{4,1}$



(a) **Manual**



(b) **Genetic**

Figure 10.8 Stored energy comparison: manual and genetic tuned systems

Leveraging automated tuning techniques can further boost system performance. Genetic algorithm methods are well-suited for systems of this framework especially if the cost function or performance score is readily available through simulation. It is important to consider constraints on the evolutionary processes underlying such optimization techniques to ensure logical consistency of the tuned system.

Chapter 11: Conclusions

This dissertation presents the Hierarchical Ensembles of Autonomous Decision Systems (HEADS) framework for multiple objective autonomy across a network of distributed systems-of-systems (NDSS). HEADS is a design methodology and reasoning structure for achieving a high degree of fully-autonomous and human-in-the-loop system autonomy. The framework supports direct inspection of behavior and performance which provides explainable autonomy. The details of the HEADS framework and its design, analysis, and optimization process are among the contributions of this dissertation as are the implementation, analysis, and optimization for a notional low Earth orbit autonomous satellite network of remote sensor and ancillary capabilities.

The HEADS design is introduced in Chapter 5 with design and analysis tools in Chapters 6 and 7 respectively. The application to a representative NDSS problem begins in Chapter 8 with results and analysis spanning Chapters 9 and 10. Refer to Chapter 1 for a complete list of contributions and chapters.

Understandability and robustness are achieved through the application of multiple linguistic variable-based fuzzy logic systems. Each expert system performs as a member in an ensemble that combines outputs to leverage per-system domains of expertise.

11.1 Explainable Autonomy Is Valuable

As networks of systems grow in size and complexity, the ability to operate autonomously becomes imperative to efficient and robust network performance. When these large complex systems incorporate humans the requirements for autonomy shift to include safety, security, and explainability. A human-in-the-loop autonomous system without explainable or verifiable behavior is a liability at best. Real-world problems driving this requirement include autonomous ground vehicles operating on public motorways and human-rated interplanetary spacecraft, among others.

The HEADS framework supports explainability through the implementation of fuzzy logic experts as the core reasoning experts. Fuzzy systems based on linguistic variables offer straightforward rule design and analysis without the need to translate or transcribe variables into meaning. Introspection of fuzzy systems is likewise direct through observation and analysis of fuzzy set and rule activation and the impact on system output generation.

11.2 HEADS Is Designer-Friendly

The HEADS design process incorporates a simple sequence:

1. signal identification based on available data and desired decision outputs,
2. rule design mapping signals to outputs, and
3. fuzzy system definition through parameterizing fuzzy set activation functions to subdivide signal domains.

This process is designed to be iterative and collaborative. It can easily scale across complex systems through subsystem-focused ensemble and expert development leveraging per-subsystem engineers and subject matter experts.

Design introspection tools are provided to measure complexity. A system designer can use these to identify layers in the hierarchy that may require additional subdivision to ensure any one part does not grow too complex.

Finally, the hierarchical structure itself supports ease of design and modification through logical collection of signals and rules. Each expert in an ensemble is designed and trained on a subset of the signal space to provide one output. The other experts in the same ensemble consider different signal space subsets or different rule paradigms.

11.3 HEADS Offers Robust Performance

Explainability and ease of design should not come at the cost of performance. The HEADS framework is proven to be performant and robust through application to a complex low Earth orbit satellite network simulation problem. The 185 simulated sensor node satellites spread across four different orbits with varying lighting and ground target access dynamics is itself a complex problem for scheduling and resource management. However, the same HEADS system design (ruleset and fuzzy set parameterization) applied to each independent satellite drove the system to meeting primary objectives and maximizing utility through secondary objectives.

Being able to design a single autonomous system and apply it to different vehicles in different environments helps reduce development and verification costs. Furthermore, because the system design is language-based and understandable it is straightforward to adapt the system, further reducing the investment required to achieve increased performance.

11.4 Performance Metrics Drive Performance

Although this statement appears tautological, signals and behaviors selected for measurement drive system design and adaptation. Manual inspection of system performance based on scoring is useful for identifying suboptimal behavior for example unexpected device activation. The importance of careful scoring design is particularly true for automated optimization since the algorithm seeks parameter sets that maximize the numerical score. If the scoring mechanism includes unimportant or irrelevant metrics the resulting system will be optimized to achieve irrelevant goals.

11.5 Future Work

Of the myriad adaptations and applications that could possibly make the HEADS framework better, a few stand out as potentially high return-on-investment approaches:

- implementing a mixed-expert HEADS system to leverage strengths of different models,
- adding a pre-weighting parameter to expert outputs based on past performance, and
- creating an ensemble-level expert prediction model to estimate which expert will perform the best based on the current signal space.

The HEADS framework as presented leverages fuzzy logic experts for explainability. Other models or expert types may be well-suited to different problem areas. Experts or models that provide an output signal with some measure of confidence or (un)certainty could be added to an ensemble. For example, hierarchical adaptive Kalman filtering has proven beneficial for adapting to unknown operating environments [88]. This approach could enable incorporating existing models and tools to the reasoning architecture to leverage prior investment and expertise.

The per-expert pre-weighting parameter is intended to perform additional gating based on past expert performance. The notional system tracks how closely each expert within an ensemble agrees with its peers. Output decisions are also compared to expected outputs providing *a posteriori* performance metrics. These two meta-parameters can be combined to identify experts within an ensemble that consistently underperform compared to peers. Then, without requiring the entire system to be modified to fix such underperforming experts, the pre-weighting parameter reduces the impact on an ensemble's output.

The pre-weighting approach could be further improved by creating an expert prediction model that attempts to identify the most-impactful expert before reasoning is performed, or *a priori*. This is of potential interest as it could provide metadata about ensemble performance and HEADS structural design for supporting future modifications. The prediction model paired with the *a posteriori* performance metrics combine to bring the HEADS framework toward a traditional recursive filtering system which could enable other interesting functions such as input signal rejection or underweighting.

References

[1] G. E. Moore, "Cramming more components onto integrated circuits," *Electronics Magazine*, vol. 38, no. 8, p. 4, 1965.

[2] M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things and Industry 4.0," *IEEE Industrial Electronics Magazine*, vol. 11, pp. 17–27, Mar. 2017. Conference Name: IEEE Industrial Electronics Magazine.

[3] M. Fisher, V. Mascardi, K. Y. Rozier, B.-H. Schlingloff, M. Winikoff, and N. Yorke-Smith, "Towards a framework for certification of reliable autonomous systems," *Autonomous Agents and Multi-Agent Systems*, vol. 35, p. 8, Dec. 2020.

[4] K. E. Schaefer, R. W. Brewer, J. Putney, E. Mottern, J. Barghout, and E. R. Straub, "Relinquishing Manual Control: Collaboration Requires the Capability to Understand Robot Intent," in *2016 International Conference on Collaboration Technologies and Systems (CTS)*, pp. 359–366, Oct. 2016.

[5] A. M. Turing, "I.—COMPUTING MACHINERY AND INTELLIGENCE," *Mind*, vol. LIX, pp. 433–460, Oct. 1950.

[6] L. A. Dennis and M. Fisher, "Verifiable Self-Aware Agent-Based Autonomous Systems," *Proceedings of the IEEE*, vol. 108, pp. 1011–1026, July 2020.

[7] M. Wooldridge and N. R. Jennings, "Intelligent agents: theory and practice," *The Knowledge Engineering Review*, vol. 10, pp. 115–152, June 1995. Publisher: Cambridge University Press.

[8] P. Radanliev, D. De Roure, R. Nicolescu, M. Huth, and O. Santos, "Artificial Intelligence and the Internet of Things in Industry 4.0," *CCF Transactions on Pervasive Computing and Interaction*, vol. 3, pp. 329–338, Sept. 2021.

[9] L. An, V. Grimm, A. Sullivan, B. L. Turner II, N. Malleson, A. Heppenstall, C. Vincenot, D. Robinson, X. Ye, J. Liu, E. Lindkvist, and W. Tang, "Challenges, tasks, and opportunities in modeling agent-based complex systems," *Ecological Modelling*, vol. 457, p. 109685, Oct. 2021.

[10] F. Figueroa, L. Underwood, and D. Armstrong, "Model-Based Systems Engineering, Real-Time Operations, and Autonomy," in *ASCEND 2020*, (Virtual Event), American Institute of Aeronautics and Astronautics, Nov. 2020.

[11] S. D. Holcomb, W. K. Porter, S. V. Ault, G. Mao, and J. Wang, "Overview on DeepMind and Its AlphaGo Zero AI," in *Proceedings of the 2018 International Conference on Big Data and Education*, (Honolulu HI USA), pp. 67–71, ACM, Mar. 2018.

[12] Y. Chen, A. Huang, Z. Wang, I. Antonoglou, J. Schrittwieser, D. Silver, and N. de Freitas, "Bayesian Optimization in AlphaGo," 2018. Publisher: arXiv Version Number: 1.

[13] Q. Al-Tashi, H. Rais, and S. J. Abdulkadir, "Hybrid Swarm Intelligence Algorithms with Ensemble Machine Learning for Medical Diagnosis," in *2018 4th International Conference on Computer and Information Sciences (ICCOINS)*, pp. 1–6, Aug. 2018.

[14] L. Blakely, M. J. Reno, and R. J. Broderick, "Decision tree ensemble machine learning for rapid QSTS simulations," in *2018 IEEE Power Energy Society Innovative Smart Grid Technologies Conference (ISGT)*, pp. 1–5, Feb. 2018. ISSN: 2472-8152.

[15] S. Fuertes, G. Picart, J.-Y. Tourneret, L. Chaari, A. Ferrari, and C. Richard, "Improving Spacecraft Health Monitoring with Automatic Anomaly Detection Techniques," in *SpaceOps 2016 Conference*, (Daejeon, Korea), American Institute of Aeronautics and Astronautics, May 2016.

[16] X. Hao, W. Ren, R. Xiong, T. Zhu, and K.-K. R. Choo, "Asymmetric cryptographic functions based on generative adversarial neural networks for Internet of Things," *Future Generation Computer Systems*, vol. 124, pp. 243–253, Nov. 2021.

[17] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining Explanations: An Overview of Interpretability of Machine Learning," in *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, pp. 80–89, Oct. 2018.

[18] F. Bacon, L. Jardine, and M. Silverthorne, *The New Organon*. Cambridge texts in the history of philosophy, Cambridge [U.K.] ; New York: Cambridge University Press, 2000.

[19] P. McCorduck, *Machines Who Think: A Personal Inquiry Into the History and Prospects of Artificial Intelligence*. Natick, Mass: A.K. Peters, 25th anniversary update ed., 2004.

[20] B. J. Copeland, "The Modern History of Computing," in *The Stanford Encyclopedia of Philosophy* (E. N. Zalta, ed.), Metaphysics Research Lab, Stanford University, winter 2020 ed., 2020.

[21] J. Fuegi and J. Francis, "Lovelace and Babbage and the Creation of the 1843 'Notes'," *IEEE Annals of the History of Computing*, vol. 25, pp. 16–26, Oct. 2003. Conference Name: IEEE Annals of the History of Computing.

[22] J. Von Neumann and O. Morgenstern, *Theory of Games and Economic Behavior*. Princeton Classic Editions, Princeton, N.J. ; Woodstock: Princeton University Press, 60th anniversary ed ed., 2007. OCLC: ocm78989126.

[23] A. L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers," *IBM Journal of Research and Development*, vol. 3, pp. 210–229, July 1959. Conference Name: IBM Journal of Research and Development.

[24] S. Veres, “Principles, architectures and trends in autonomous control,” in *2005 The IEE Seminar on Autonomous Agents in Control (Ref. No. 2005/10986)*, pp. 1–9, May 2005. ISSN: 0537-9989.

[25] E. Eberbach, “Approximate reasoning in the algebra of bounded rational agents,” *International Journal of Approximate Reasoning*, vol. 49, pp. 316–330, Oct. 2008.

[26] R. A. Brooks, “Elephants don’t play chess,” *Robotics and Autonomous Systems*, vol. 6, pp. 3–15, June 1990.

[27] R. Brooks, “A robust layered control system for a mobile robot,” *IEEE Journal on Robotics and Automation*, vol. 2, no. 1, pp. 14–23, 1986.

[28] P. Ridao, M. Carreras, J. Batlle, and J. Amat, “O2CA2: A New Hybrid Control Architecture for a Low Cost AUV,” *IFAC Proceedings Volumes*, vol. 34, pp. 311–316, July 2001.

[29] B. Hayes-Roth, “A blackboard architecture for control,” *Artificial Intelligence*, vol. 26, pp. 251–321, July 1985.

[30] C. Flanagan, D. Toal, and M. Leyden, “Subsumption and Fuzzy-Logic, Experiments in Behavior-Based Control of Mobile Robots,” *International Journal of Smart Engineering System Design*, vol. 5, pp. 161–175, July 2003.

[31] M. P. Georgeff and A. L. Lansky, “Reactive reasoning and planning,” in *Proceedings of the sixth National conference on Artificial intelligence - Volume 2*, AAAI’87, (Seattle, Washington), pp. 677–682, AAAI Press, July 1987.

[32] J. P. Müller, M. Pischel, and M. Thiel, “Modeling reactive behaviour in vertically layered agent architectures,” in *Intelligent Agents* (M. J. Wooldridge and N. R. Jennings, eds.), Lecture Notes in Computer Science, (Berlin, Heidelberg), pp. 261–276, Springer, 1995.

[33] J. J. Bryson, *Intelligence by design : principles of modularity and coordination for engineering complex adaptive agents.* Thesis, Massachusetts Institute of Technology, 2001. Accepted: 2005-08-23T18:28:44Z.

[34] N. Abramson, “Development of the ALOHANET,” *IEEE Transactions on Information Theory*, vol. 31, pp. 119–123, Mar. 1985.

[35] H. Hemmati, “Optical systems for free-space laser communications,” in *Current Developments in Lens Design and Optical Engineering IV*, vol. 5173, pp. 64–68, SPIE, Nov. 2003.

[36] K. Wilson, M. Wright, S. Lee, and M. Troy, “Adaptive optics for daytime deep space laser communications from Mars,” in *Digest of the LEOS Summer Topical Meetings, 2005.*, pp. 19–20, July 2005. ISSN: 2376-8614.

[37] I. U. Zaman, J. E. Velazco, and O. Boyraz, "Realization of Omnidirectional CubeSat Crosslink by Wavelength-Selective Optical Transceiver," *IEEE Journal on Miniaturization for Air and Space Systems*, vol. 1, pp. 47–55, June 2020.

[38] D. Alaluf, M. Centrone, D. B. Calia, P. Suetterling, R. Speziali, M. Faccini, W. Hackenberg, J. P. Armengol, and A. D. Paola, "Paving the way to daytime optical feeder links based on LGS assisted adaptive optics," in *International Conference on Space Optics — ICSO 2020*, vol. 11852, pp. 2438–2444, SPIE, June 2021.

[39] J. Zhu, J. Zhu, Z. Wang, S. Guo, and C. Xu, "Hierarchical Decision and Control for Continuous Multitarget Problem: Policy Evaluation With Action Delay," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, pp. 464–473, Feb. 2019. Conference Name: IEEE Transactions on Neural Networks and Learning Systems.

[40] J. M. Badger, P. Strawser, and C. Claunch, "A Distributed Hierarchical Framework for Autonomous Spacecraft Control," in *2019 IEEE Aerospace Conference*, pp. 1–8, Mar. 2019. ISSN: 1095-323X.

[41] S. Riedmaier, B. Danquah, B. Schick, and F. Diermeyer, "Unified Framework and Survey for Model Verification, Validation and Uncertainty Quantification," *Archives of Computational Methods in Engineering*, vol. 28, pp. 2655–2688, June 2021.

[42] M. A. Boden, *Artificial Intelligence and Natural Man.* No. 5063 in Harper torchbooks, New York: Basic Books, 1977.

[43] T. M. Mitchell, *Machine Learning.* McGraw-Hill series in computer science, New York: McGraw-Hill, 1997.

[44] B. Dasarathy and B. Sheela, "A composite classifier system design: Concepts and methodology," *Proceedings of the IEEE*, vol. 67, no. 5, pp. 708–713, 1979. Conference Name: Proceedings of the IEEE.

[45] C. M. Bishop, *Neural Networks for Pattern Recognition.* Clarendon Press, Nov. 1995. Google-Books-ID: T0S0BgAAQBAJ.

[46] M. Markou and S. Singh, "Novelty detection: a review—part 2:: neural network based approaches," *Signal Processing*, vol. 83, pp. 2499–2521, Dec. 2003.

[47] P. V. R. Ferreira, R. Paffenroth, A. M. Wyglinski, T. M. Hackett, S. G. Bilén, R. C. Reinhart, and D. J. Mortensen, "Multi-objective reinforcement learning-based deep neural networks for cognitive space communications," in *2017 Cognitive Communications for Aerospace Applications Workshop (CCAA)*, pp. 1–8, June 2017.

[48] J. E. van Engelen and H. H. Hoos, "A survey on semi-supervised learning," *Machine Learning*, vol. 109, pp. 373–440, Feb. 2020.

[49] L. Hansen and P. Salamon, "Neural Network Ensembles," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 10, pp. 993–1001, 1990. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[50] L. Breiman, “Bagging Predictors,” *Machine Learning*, vol. 24, pp. 123–140, Aug. 1996.

[51] Y. Freund and R. E. Schapire, “Experiments with a New Boosting Algorithm,” *Proceedings of the Thirteenth International Conference on Machine Learning*, July 1996.

[52] L. Breiman, “Random Forests,” *Machine Learning*, vol. 45, pp. 5–32, Oct. 2001.

[53] D. E. Goldberg, *Genetic algorithms in search, optimization, and machine learning.* Reading, Mass: Addison-Wesley Pub. Co, 1989.

[54] J. Treboux, D. Genoud, and R. Ingold, “Decision Tree Ensemble Vs. N.N. Deep Learning: Efficiency Comparison For A Small Image Dataset,” in *2018 International Workshop on Big Data and Information Security (IWBIS)*, pp. 25–30, May 2018.

[55] X. Liu, G. Wang, Z. Cai, and H. Zhang, “A MultiBoosting Based Transfer Learning Algorithm,” *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 19, no. 3, pp. 381–388, 2015.

[56] X. Liu, Z. Liu, G. Wang, Z. Cai, and H. Zhang, “Ensemble Transfer Learning Algorithm,” *IEEE Access*, vol. 6, pp. 2389–2396, 2018. Conference Name: IEEE Access.

[57] J. R. Boyd, “The Essence of Winning and Losing,” 1996.

[58] A. Gegov, *Fuzzy Networks for Complex Systems*, vol. 259 of *Studies in Fuzziness and Soft Computing.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2011.

[59] S.-M. Zhou and J. Q. Gan, “Low-level interpretability and high-level interpretability: a unified view of data-driven interpretable fuzzy system modelling,” *Fuzzy Sets and Systems*, vol. 159, pp. 3091–3131, Dec. 2008.

[60] R. Polikar, “Ensemble based systems in decision making,” *IEEE Circuits and Systems Magazine*, vol. 6, no. 3, pp. 21–45, 2006. Conference Name: IEEE Circuits and Systems Magazine.

[61] M. Iván Tarride, “The complexity of measuring complexity,” *Kybernetes*, vol. 42, pp. 174–184, Jan. 2013. Publisher: Emerald Group Publishing Limited.

[62] M. Mantyla, “A modeling system for top-down design of assembled products,” *IBM Journal of Research and Development*, vol. 34, pp. 636–659, Sept. 1990. Conference Name: IBM Journal of Research and Development.

[63] N. Cramer, D. Cellucci, C. Adams, A. Sweet, M. Hejase, J. Frank, R. Levinson, S. Gridnev, and L. Brown, “Design and Testing of Autonomous Distributed Space Systems,” *Proceedings of the Small Satellite Conference, Technical Session 1: Mission Operations and Autonomy, 129.*, 2021.

[64] L. A. Zadeh, “Outline of a New Approach to the Analysis of Complex Systems and Decision Processes,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-3, pp. 28–44, Jan. 1973. Conference Name: IEEE Transactions on Systems, Man, and Cybernetics.

[65] L. A. Zadeh, "Fuzzy sets as a basis for a theory of possibility," *Fuzzy Sets and Systems*, vol. 1, pp. 3–28, Jan. 1978.

[66] B. Kosko, "Fuzziness VS. Probability," *International Journal of General Systems*, vol. 17, no. 2-3, pp. 211–240, 1990.

[67] J.-S. R. Jang, C.-T. Sun, and E. Mizutani, *Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence.* MATLAB curriculum series, Upper Saddle River, NJ: Prentice Hall, 1997.

[68] U. D. Hanebeck and G. K. Schmidt, "Genetic optimization of fuzzy networks," *Fuzzy Sets and Systems*, vol. 79, pp. 59–68, Apr. 1996.

[69] O. Cordón, *Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases.* World Scientific, 2001.

[70] N. Ernest, K. Cohen, E. Kivelevitch, C. Schumacher, and D. Casbeer, "Genetic Fuzzy Trees and their Application Towards Autonomous Training and Control of a Squadron of Unmanned Combat Aerial Vehicles," *Unmanned Systems*, vol. 03, pp. 185–204, July 2015. Publisher: World Scientific Publishing Co.

[71] N. D. Ernest, E. Garcia, D. Casbeer, K. Cohen, and C. J. Schumacher, "Multi-agent Cooperative Decision Making using Genetic Cascading Fuzzy Systems," in *AIAA Infotech @ Aerospace*, (Kissimmee, Florida), American Institute of Aeronautics and Astronautics, Jan. 2015.

[72] D. Dubois and H. M. Prade, *Fuzzy sets and systems: theory and applications.* No. v. 144 in Mathematics in science and engineering, New York: Academic Press, 1980.

[73] E. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *International Journal of Man-Machine Studies*, vol. 7, pp. 1–13, Jan. 1975.

[74] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-15, pp. 116–132, Jan. 1985.

[75] M. Sugeno and G. Kang, "Structure identification of fuzzy model," *Fuzzy Sets and Systems*, vol. 28, pp. 15–33, Oct. 1988.

[76] F. Bullo, F. Fagnani, and B. Franci, "Finite-Time Influence Systems and the Wisdom of Crowd Effect," *SIAM Journal on Control and Optimization*, vol. 58, pp. 636–659, Jan. 2020. Publisher: Society for Industrial and Applied Mathematics.

[77] I. D. Craig, "Blackboard systems," *Artificial Intelligence Review*, vol. 2, pp. 103–118, June 1988.

[78] P. J. Rossomando, "The achievement of spacecraft autonomy through the thematic application of multiple cooperating intelligent agents," *Telematics and Informatics*, vol. 9, pp. 205–219, June 1992.

[79] V. M. Kokorakis, M. Petridis, and S. Kapetanakis, "A Blackboard Based Hybrid Multi-Agent System for Improving Classification Accuracy Using Reinforcement Learning Techniques," in *Artificial Intelligence XXXIV* (M. Bramer and M. Petridis, eds.), Lecture Notes in Computer Science, (Cham), pp. 47–57, Springer International Publishing, 2017.

[80] R. E. Schapire, "The Strength of Weak Learnability," *Machine Learning*, vol. 5, no. 2, pp. 197–227, 1990.

[81] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive Mixtures of Local Experts," *Neural Computation*, vol. 3, pp. 79–87, Mar. 1991. Conference Name: Neural Computation.

[82] X. Gu, P. Angelov, and Z. Zhao, "Self-organizing fuzzy inference ensemble system for big streaming data classification," *Knowledge-Based Systems*, vol. 218, p. 106870, Apr. 2021.

[83] J. M. Cadenas, M. C. Garrido, A. Martínez, and R. Martínez, "Consensus operators for decision making in Fuzzy Random Forest ensemble," in *2011 11th International Conference on Intelligent Systems Design and Applications*, pp. 1377–1382, Nov. 2011. ISSN: 2164-7151.

[84] F. Prado, M. C. Minutolo, and W. Kristjanpoller, "Forecasting based on an ensemble Autoregressive Moving Average - Adaptive neuro - Fuzzy inference system – Neural network - Genetic Algorithm Framework," *Energy*, vol. 197, p. 117159, Apr. 2020.

[85] C. Wu, C. Lin, D. Barnes, and Y. Zhang, "Partner selection in sustainable supply chains: A fuzzy ensemble learning model," *Journal of Cleaner Production*, vol. 275, p. 123165, Dec. 2020.

[86] M. I. Jordan and R. A. Jacobs, "Hierarchies of adaptive experts," in *Proceedings of the 4th International Conference on Neural Information Processing Systems*, NIPS'91, (San Francisco, CA, USA), pp. 985–992, Morgan Kaufmann Publishers Inc., Dec. 1991.

[87] M. I. Jordan and R. A. Jacobs, "Hierarchical Mixtures of Experts and the EM Algorithm," *Neural Computation*, vol. 6, pp. 181–214, Mar. 1994. Conference Name: Neural Computation.

[88] W. Chaer, R. Bishop, and J. Ghosh, "Hierarchical adaptive Kalman filtering for interplanetary orbit determination," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 34, pp. 883–896, July 1998. Conference Name: IEEE Transactions on Aerospace and Electronic Systems.

[89] G. V. S. Raju, J. Zhou, and R. A. Kisner, "Hierarchical fuzzy control," *International Journal of Control*, vol. 54, pp. 1201–1216, Nov. 1991. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/00207179108934205.

[90] R. Yager, “On a hierarchical structure for fuzzy modeling and control,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, pp. 1189–1197, July 1993. Conference Name: IEEE Transactions on Systems, Man, and Cybernetics.

[91] R. Yager, “On the construction of hierarchical fuzzy systems models,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 28, pp. 55–66, Feb. 1998. Conference Name: IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews).

[92] S. Aja-FernÁndez and C. Alberola-LÓpez, “Matrix Modeling of Hierarchical Fuzzy Systems,” *IEEE Transactions on Fuzzy Systems*, vol. 16, pp. 585–599, June 2008. Conference Name: IEEE Transactions on Fuzzy Systems.

[93] J. Yerushalmy, “Statistical Problems in Assessing Methods of Medical Diagnosis, with Special Reference to X-Ray Techniques,” *Public Health Reports (1896-1970)*, vol. 62, no. 40, p. 1432, 1947.

[94] C. E. Metz, “Basic principles of ROC analysis,” *Seminars in Nuclear Medicine*, vol. 8, pp. 283–298, Oct. 1978.

[95] J. L. Wildman, E. Salas, and C. P. R. Scott, “Measuring Cognition in Teams: A Cross-Domain Review,” *Human Factors*, vol. 56, pp. 911–941, Aug. 2014. Publisher: SAGE Publications Inc.

[96] S. Jiang and R. C. Arkin, “Mixed-Initiative Human-Robot Interaction: Definition, Taxonomy, and Survey,” in *2015 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 954–961, Oct. 2015.

[97] A. Bicchi, “Of Robots, Humans, Bodies and Intelligence: Body Languages for Human Robot Interaction,” in *2015 10th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 1–1, Mar. 2015. ISSN: 2167-2121.

[98] C. Xue, Y. Qiao, and N. Murray, “Enabling Human-Robot-Interaction for Remote Robotic Operation via Augmented Reality,” in *2020 IEEE 21st International Symposium on ”A World of Wireless, Mobile and Multimedia Networks” (WoWMoM)*, pp. 194–196, Aug. 2020.

[99] P. H. Kahn, B. T. Gill, A. L. Reichert, T. Kanda, H. Ishiguro, and J. H. Ruckert, “Validating interaction patterns in HRI,” in *2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 183–184, Mar. 2010. ISSN: 2167-2148.

[100] M. Abdullah, M. Ahmad, and D. Han, “Hierarchical Attention Approach in Multimodal Emotion Recognition for Human Robot Interaction,” in *2021 36th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC)*, pp. 1–4, June 2021.

[101] J. H. Ruckert, P. H. Kahn, T. Kanda, H. Ishiguro, S. Shen, and H. E. Gary, “Designing for sociality in HRI by means of multiple personas in robots,” in *2013 8th ACM/IEEE*

International Conference on Human-Robot Interaction (HRI), pp. 217–218, Mar. 2013. ISSN: 2167-2148.

[102] K. Malchus, P. Stenneken, P. Jaecks, C. Meyer, O. Damm, and B. Wrede, "The role of emotional congruence in human-robot interaction," in *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 191–192, Mar. 2013. ISSN: 2167-2148.

[103] S. Pinker, *How the Mind Works.* Norton, Jan. 1997. Google-Books-ID: fBaqQgAACAAJ.

[104] Z. Zong and C. Hong, "On Application of Natural Language Processing in Machine Translation," in *2018 3rd International Conference on Mechanical, Control and Computer Engineering (ICMCCE)*, pp. 506–510, Sept. 2018.

[105] H. Isahara, "Resource-based Natural Language Processing," in *2007 International Conference on Natural Language Processing and Knowledge Engineering*, pp. 11–12, Aug. 2007.

[106] M. Ahirrao, Y. Joshi, A. Gandhe, S. Kotgire, and R. G. Deshmukh, "Phrase Composing Tool using Natural Language Processing," in *2021 International Conference on Intelligent Technologies (CONIT)*, pp. 1–4, June 2021.

[107] M. Bates, "Models of natural language understanding.," *Proceedings of the National Academy of Sciences*, vol. 92, pp. 9977–9982, Oct. 1995. Publisher: Proceedings of the National Academy of Sciences.

[108] D. W. Otter, J. R. Medina, and J. K. Kalita, "A Survey of the Usages of Deep Learning for Natural Language Processing," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, pp. 604–624, Feb. 2021. Conference Name: IEEE Transactions on Neural Networks and Learning Systems.

[109] N. Camelin, F. Bechet, G. Damnati, and R. De Mori, "Detection and Interpretation of Opinion Expressions in Spoken Surveys," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, pp. 369–381, Feb. 2010. Conference Name: IEEE Transactions on Audio, Speech, and Language Processing.

[110] D. U. Patton, W. R. Frey, K. A. McGregor, F.-T. Lee, K. McKeown, and E. Moss, "Contextual Analysis of Social Media: The Promise and Challenge of Eliciting Context in Social Media Posts with Natural Language Processing," in *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pp. 337–342, New York, NY, USA: Association for Computing Machinery, Feb. 2020.

[111] A. Galassi, M. Lippi, and P. Torroni, "Attention in Natural Language Processing," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, pp. 4291–4308, Oct. 2021. Conference Name: IEEE Transactions on Neural Networks and Learning Systems.

[112] J. Deng and F. Ren, "A Survey of Textual Emotion Recognition and Its Challenges," *IEEE Transactions on Affective Computing*, pp. 1–1, 2021. Conference Name: IEEE Transactions on Affective Computing.

[113] S. Yang and B. Bhanu, "Understanding Discrete Facial Expressions in Video Using an Emotion Avatar Image," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, pp. 980–992, Aug. 2012. Conference Name: IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics).

[114] S. Ebrahimi Kahou, V. Michalski, K. Konda, R. Memisevic, and C. Pal, "Recurrent Neural Networks for Emotion Recognition in Video," in *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*, ICMI '15, (New York, NY, USA), pp. 467–474, Association for Computing Machinery, Nov. 2015.

[115] F. Noroozi, M. Marjanovic, A. Njegus, S. Escalera, and G. Anbarjafari, "Audio-Visual Emotion Recognition in Video Clips," *IEEE Transactions on Affective Computing*, vol. 10, pp. 60–75, Jan. 2019. Conference Name: IEEE Transactions on Affective Computing.

[116] J. Shu, M. Chiu, and P. Hui, "Emotion Sensing for Mobile Computing," *IEEE Communications Magazine*, vol. 57, pp. 84–90, Nov. 2019. Conference Name: IEEE Communications Magazine.

[117] S. Zhao, S. Wang, M. Soleymani, D. Joshi, and Q. Ji, "Affective Computing for Large-scale Heterogeneous Multimedia Data: A Survey," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 15, pp. 93:1–93:32, Dec. 2019.

[118] O. Cordón, "A historical review of evolutionary learning methods for Mamdani-type fuzzy rule-based systems: Designing interpretable genetic fuzzy systems," *International Journal of Approximate Reasoning*, vol. 52, pp. 894–913, Sept. 2011.

[119] D. Nauck and R. Kruse, "How the learning of rule weights affects the interpretability of fuzzy systems," in *1998 IEEE International Conference on Fuzzy Systems Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98CH36228)*, vol. 2, pp. 1235–1240 vol.2, May 1998. ISSN: 1098-7584.

[120] J. M. Alonso, L. Magdalena, and S. Guillaume, "HILK: A new methodology for designing highly interpretable linguistic knowledge bases using the fuzzy logic formalism," *International Journal of Intelligent Systems*, vol. 23, no. 7, pp. 761–794, 2008. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/int.20288.

[121] L. Hong, P. Lamberson, and S. E. Page, "Hybrid Predictive Ensembles: Synergies Between Human and Computational Forecasts," *Journal of Social Computing*, vol. 2, pp. 89–102, June 2021. Conference Name: Journal of Social Computing.

[122] J. S. Bridle, "Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition," in *Neurocomputing* (F. F. Soulié and J. Hérault, eds.), NATO ASI Series, (Berlin, Heidelberg), pp. 227–236, Springer, 1990.

[123] Y. Kozai, "New determination of zonal harmonics coeffi-cients of the earth's gravitational potential special report no. 165," tech. rep., 1964.

[124] V. G. Szebehely, *Adventures in Celestial Mechanics.* 1989. Publication Title: Austin ADS Bibcode: 1989acmf.book.....S.

Appendix A: Example HEADS Definitions and Parameterizations

This appendix contains the example problem system designs including the baseline rules and parameters in Table A.1 and Table A.2 respectively.

The manually-tuned rules and parameters are in Table A.3 and Table A.4 and only show the differences compared to the baseline system.

Finally, the parameters adapted by genetic algorithm are shown in Table A.5. The genetic algorithm tuning did not modify any rules compared to the manually-tuned system.

Table A.1 HEADS example - baseline system rule definitions

y_j	$s_{j,p}$	Rule Index	Rule Definition
y_1	$s_{1,1}$	01	batt_critical THEN self_bad
		02	batt_low AND dt_sun_long THEN self_bad
		03	batt_low AND (dt_sun_short OR sunlit) THEN self_marginal
		04	batt_good AND dt_sun_long THEN self_marginal
		05	batt_good AND (dt_sun_short OR sunlit) THEN self_good
		06	batt_full THEN self_good
	$s_{1,2}$	01	faulty AND batt_low THEN self_bad
		02	faulty AND batt_good THEN self_marginal
		03	NOT faulty THEN self_good
		04	fault_recovery THEN self_bad
y_2	$s_{2,1}$	01	batt_low THEN sensor_off
		02	batt_good AND (NOT sunlit OR dt_sun_long) THEN sensor_off
		03	batt_good AND (sunlit OR dt_sun_short) THEN sensor_on
		04	batt_full THEN sensor_on

Table continues on next page

Table A.1 (continued)

Ensemble	Expert	Rule Index	Rule Definition
	$s_{2,2}$	01	NOT target_visible THEN sensor_off
		02	dt_target_long THEN sensor_off
		03	dt_target_short THEN sensor_on
		04	target_visible THEN sensor_on
		05	sensor_1_active OR sensor_2_active OR sensor_3_active THEN sensor_on
	$s_{2,3}$	01	sn_count_low THEN sensor_on
		02	sn_count_high THEN sensor_off
		03	sn_good_low OR sn_bad_high THEN sensor_on
		04	sn_good_high OR sn_bad_low THEN sensor_off
	$s_{2,4}$	01	faulty THEN sensor_off
		02	NOT faulty THEN sensor_on
		03	fault_recovery THEN sensor_off
		04	NOT self_good THEN sensor_off
y_3	$s_{3,1}$	01	batt_low THEN sensor_off
		02	batt_good AND (NOT sunlit OR dt_sun_long) THEN sensor_off
		03	batt_good AND (sunlit OR dt_sun_short) THEN sensor_on
		04	batt_full THEN sensor_on
	$s_{3,2}$	01	NOT target_visible THEN sensor_off
		02	dt_target_long THEN sensor_off
		03	dt_target_short THEN sensor_on
		04	target_visible THEN sensor_on
		05	sensor_1_active OR sensor_2_active OR sensor_3_active THEN sensor_on
	$s_{3,3}$	01	sn_count_low THEN sensor_on
		02	sn_count_high THEN sensor_off
		03	sn_good_low OR sn_bad_high THEN sensor_on
		04	sn_good_high OR sn_bad_low THEN sensor_off

Table continues on next page

Table A.1 (continued)

Ensemble	Expert	Rule Index	Rule Definition
	$s_{3,4}$	01	faulty THEN sensor_off
		02	NOT faulty THEN sensor_on
		03	fault_recovery THEN sensor_off
		04	NOT self_good THEN sensor_off
y_4	$s_{4,1}$	01	batt_low THEN sensor_off
		02	batt_good AND (NOT sunlit OR dt_sun_long) THEN sensor_off
		03	batt_good AND (sunlit OR dt_sun_short) THEN sensor_on
		04	batt_full THEN sensor_on
	$s_{4,2}$	01	NOT target_visible THEN sensor_off
		02	dt_target_long THEN sensor_off
		03	dt_target_short THEN sensor_on
		04	target_visible THEN sensor_on
		05	sensor_1_active OR sensor_2_active OR sensor_3_active THEN sensor_on
	$s_{4,3}$	01	sn_count_low THEN sensor_on
		02	sn_count_high THEN sensor_off
		03	sn_good_low OR sn_bad_high THEN sensor_on
		04	sn_good_high OR sn_bad_low THEN sensor_off
	$s_{4,4}$	1	faulty THEN sensor_off
		02	NOT faulty THEN sensor_on
		03	fault_recovery THEN sensor_off
		04	NOT self_good THEN sensor_off
y_5	$s_{5,1}$	01	batt_bad THEN dtx_off
		02	batt_bad AND ((dt_sun_short OR sunlit) AND (COM_neighbors OR PROC_neighbors)) THEN dtx_on
		03	batt_good AND (COM_neighbors OR PROC_neighbors) THEN dtx_on

Table continues on next page

Table A.1 (continued)

Ensemble	Expert	Rule Index	Rule Definition
		04	batt_good AND (dt_sun_long AND (COM_neighbors OR PROC_neighbors)) THEN dtx_off
		05	(sensor_1_active OR sensor_2_active OR sensor_3_active) AND (COM_neighbors OR PROC_neighbors) THEN dtx_on
	$s_{5,2}$	01	faulty OR fault_recovery THEN dtx_off
		02	NOT faulty AND NOT fault_recovery THEN dtx_on
		03	NOT self_good THEN dtx_off
y_6	$s_{6,1}$	01	NOT faulty THEN fault_recovery_off
		02	batt_critical THEN fault_recovery_on
		03	NOT batt_critical THEN fault_recovery_off
		04	(faulty OR fault_recovery) THEN fault_recovery_on
		05	NOT faulty AND NOT fault_recovery THEN fault_recovery_off

Table A.2 HEADS example - baseline system fuzzy set parameters

y_j	$s_{j,p}$	Antecedent/ Consequent	Fuzzy Set	Activ. Func.	Parameters	Signal
y_1	$s_{1,1}$	antecedent	batt_critical	ltrap	[10, 20]	x_1: stored_energy
			batt_low	trap	[15, 20, 25, 30]	x_1: stored_energy
			batt_good	trap	[30, 50, 70, 80]	x_1: stored_energy
			batt_full	rtrap	[60, 80]	x_1: stored_energy
			sunlit	bool	[ ]	x_2: sunlit
			dt_sun_short	ltrap	[300, 600]	x_3: dt_sun_s
			dt_sun_long	rtrap	[300, 900]	x_3: dt_sun_s

Table continues on next page

Table A.2 (continued)

y_j	$s_{j,p}$	Antecedent/ Consequent	Fuzzy Set	Activ. Func.	Parameters	Signal
		consequent	self_bad	ltrap	[0.4, 0.5]	y_1: self_good
			self_marginal	trap	[0.45, 0.5, 0.55, 0.65]	y_1: self_good
			self_good	rtrap	[0.6, 0.7]	y_1: self_good
	$s_{1,2}$	antecedent	batt_low	ltrap	[40, 60]	x_1: stored_energy
			batt_good	rtrap	[50, 70]	x_1: stored_energy
			faulty	bool	[ ]	x_{11}: fault_detected
			fault_recovery	bool	[ ]	y_6: fault_recovery
		consequent	self_bad	ltrap	[0.3, 0.4]	y_1: self_good
			self_marginal	trap	[0.45, 0.5, 0.55, 0.6]	y_1: self_good
			self_good	rtrap	[0.55, 0.7]	y_1: self_good
y_2	$s_{2,1}$	antecedent	batt_low	ltrap	[30, 40]	x_1: stored_energy
			batt_good	tri	[30, 60]	x_1: stored_energy
			batt_full	rtrap	[50, 60]	x_1: stored_energy
			sunlit	bool	[ ]	x_2: sunlit
			dt_sun_short	ltrap	[180, 300]	x_3: dt_sun_s
			dt_sun_long	rtrap	[240, 480]	x_3: dt_sun_s
		consequent	sensor_off	ltrap	[0.4, 0.55]	y_2: sensor_1_active
			sensor_on	rtrap	[0.45, 0.6]	y_2: sensor_1_active
	$s_{2,2}$	antecedent	target_visible	bool	[ ]	x_4: target_visible
			dt_target_short	ltrap	[300, 600]	x_5: dt_target_s
			dt_target_long	rtrap	[480, 780]	x_5: dt_target_s
			sensor_1_active	bool	[ ]	y_2: sensor_1_active
			sensor_2_active	bool	[ ]	y_3: sensor_2_active
			sensor_3_active	bool	[ ]	y_4: sensor_3_active

Table continues on next page

Table A.2 (continued)

y_j	$s_{j,p}$	Antecedent/ Consequent	Fuzzy Set	Activ. Func.	Parameters	Signal
		consequent	sensor_off	ltrap	[0.4, 0.55]	y_2: sensor_1_active
			sensor_on	rtrap	[0.45, 0.6]	y_2: sensor_1_active
	$s_{2,3}$	antecedent	sn_count_low	ltrap	[5, 7]	x_8: SEN_neighbors_count
			sn_count_high	rtrap	[6, 8]	x_8: SEN_neighbors_count
			sn_good_low	ltrap	[2, 3]	x_9: SEN_neighbors_good
			sn_good_high	rtrap	[2, 3]	x_9: SEN_neighbors_good
			sn_bad_low	ltrap	[3, 4]	x_{10}: SEN_neighbors_bad
			sn_bad_high	rtrap	[3, 4]	x_{10}: SEN_neighbors_bad
		consequent	sensor_off	ltrap	[0.2, 0.4]	y_2: sensor_1_active
			sensor_on	rtrap	[0.3, 0.5]	y_2: sensor_1_active
	$s_{2,4}$	antecedent	faulty	bool	[ ]	x_{11}: fault_detected
			fault_recovery	bool	[ ]	y_6: fault_recovery
			self_good	bool	[ ]	y_1: self_good
		consequent	sensor_off	ltrap	[0.5, 0.6]	y_2: sensor_1_active
			sensor_on	rtrap	[0.5, 0.6]	y_2: sensor_1_active
y_3	$s_{3,1}$	antecedent	batt_low	ltrap	[40, 60]	x_1: stored_energy
			batt_good	tri	[55, 65]	x_1: stored_energy
			batt_full	rtrap	[60, 70]	x_1: stored_energy
			sunlit	bool	[ ]	x_2: sunlit
			dt_sun_short	ltrap	[60, 180]	x_3: dt_sun_s
			dt_sun_long	rtrap	[120, 240]	x_3: dt_sun_s
		consequent	sensor_off	ltrap	[0.5, 0.6]	y_3: sensor_2_active
			sensor_on	rtrap	[0.5, 0.6]	y_3: sensor_2_active
	$s_{3,2}$	antecedent	target_visible	bool	[ ]	x_4: target_visible
			dt_target_short	ltrap	[60, 180]	x_5: dt_target_s
			dt_target_long	rtrap	[120, 300]	x_5: dt_target_s
			sensor_1_active	bool	[ ]	y_2: sensor_1_active

Table continues on next page

Table A.2 (continued)

y_j	$s_{j,p}$	Antecedent/ Consequent	Fuzzy Set	Activ. Func.	Parameters	Signal
			sensor_2_active	bool	[ ]	y_3: sensor_2_active
			sensor_3_active	bool	[ ]	y_4: sensor_3_active
		consequent	sensor_off	ltrap	[0.4, 0.6]	y_3: sensor_2_active
			sensor_on	rtrap	[0.5, 0.6]	y_3: sensor_2_active
	$s_{3,3}$	antecedent	sn_count_low	ltrap	[7, 8]	x_8: SEN_neighbors_count
			sn_count_high	rtrap	[7, 8]	x_8: SEN_neighbors_count
			sn_good_low	ltrap	[4, 5]	x_9: SEN_neighbors_good
			sn_good_high	rtrap	[4, 5]	x_9: SEN_neighbors_good
			sn_bad_low	ltrap	[1, 2]	x_{10}: SEN_neighbors_bad
			sn_bad_high	rtrap	[1, 2]	x_{10}: SEN_neighbors_bad
		consequent	sensor_off	ltrap	[0.3, 0.4]	y_3: sensor_2_active
			sensor_on	rtrap	[0.35, 0.45]	y_3: sensor_2_active
	$s_{3,4}$	antecedent	faulty	bool	[ ]	x_{11}: fault_detected
			fault_recovery	bool	[ ]	y_6: fault_recovery
			self_good	bool	[ ]	y_1: self_good
		consequent	sensor_off	ltrap	[0.3, 0.4]	y_3: sensor_2_active
			sensor_on	rtrap	[0.35, 0.7]	y_3: sensor_2_active
y_4	$s_{4,1}$	antecedent	batt_low	ltrap	[50, 60]	x_1: stored_energy
			batt_good	tri	[55, 70]	x_1: stored_energy
			batt_full	rtrap	[60, 80]	x_1: stored_energy
			sunlit	bool	[ ]	x_2: sunlit
			dt_sun_short	ltrap	[60, 120]	x_3: dt_sun_s
			dt_sun_long	rtrap	[60, 120]	x_3: dt_sun_s
		consequent	sensor_off	ltrap	[0.5, 0.6]	y_4: sensor_3_active
			sensor_on	rtrap	[0.5, 0.6]	y_4: sensor_3_active
	$s_{4,2}$	antecedent	target_visible	bool	[ ]	x_4: target_visible
			dt_target_short	ltrap	[60, 120]	x_5: dt_target_s

Table continues on next page

Table A.2 (continued)

y_j	$s_{j,p}$	Antecedent/ Consequent	Fuzzy Set	Activ. Func.	Parameters	Signal
			dt_target_long	rtrap	[60, 120]	x_5: dt_target_s
			sensor_1_active	bool	[ ]	y_2: sensor_1_active
			sensor_2_active	bool	[ ]	y_3: sensor_2_active
			sensor_3_active	bool	[ ]	y_4: sensor_3_active
		consequent	sensor_off	ltrap	[0.3, 0.4]	y_4: sensor_3_active
			sensor_on	rtrap	[0.6, 0.7]	y_4: sensor_3_active
	$s_{4,3}$	antecedent	sn_count_low	ltrap	[6, 8]	x_8: SEN_neighbors_count
			sn_count_high	rtrap	[7, 8]	x_8: SEN_neighbors_count
			sn_good_low	ltrap	[3, 5]	x_9: SEN_neighbors_good
			sn_good_high	rtrap	[4, 6]	x_9: SEN_neighbors_good
			sn_bad_low	ltrap	[1, 3]	x_{10}: SEN_neighbors_bad
			sn_bad_high	rtrap	[2, 3]	x_{10}: SEN_neighbors_bad
		consequent	sensor_off	ltrap	[0.3, 0.4]	y_4: sensor_3_active
			sensor_on	rtrap	[0.35, 0.45]	y_4: sensor_3_active
	$s_{4,4}$	antecedent	faulty	bool	[ ]	x_{11}: fault_detected
			fault_recovery	bool	[ ]	y_6: fault_recovery
			self_good	bool	[ ]	y_1: self_good
		consequent	sensor_off	ltrap	[0.3, 0.4]	y_4: sensor_3_active
			sensor_on	rtrap	[0.5, 0.6]	y_4: sensor_3_active
y_5	$s_{5,1}$	antecedent	batt_bad	ltrap	[30, 60]	x_1: stored_energy
			batt_good	rtrap	[30, 60]	x_1: stored_energy
			sunlit	bool	[ ]	x_2: sunlit
			dt_sun_short	ltrap	[300, 540]	x_3: dt_sun_s
			dt_sun_long	rtrap	[420, 600]	x_3: dt_sun_s
			COM_neighbors	rtrap	[0, 1]	x_6: COM_neighbors_count
			PROC_neighbors	rtrap	[0, 1]	x_7: PROC_neighbors_count
			sensor_1_active	bool	[ ]	y_2: sensor_1_active

Table continues on next page

Table A.2 (continued)

y_j	$s_{j,p}$	Antecedent/ Consequent	Fuzzy Set	Activ. Func.	Parameters	Signal
			sensor_2_active	bool	[ ]	y_3: sensor_2_active
			sensor_3_active	bool	[ ]	y_4: sensor_3_active
		consequent	dtx_off	ltrap	[0.4, 0.6]	y_5: dtx_active
			dtx_on	rtrap	[0.4, 0.6]	y_5: dtx_active
	$s_{5,2}$	antecedent	faulty	bool	[ ]	x_{11}: fault_detected
			fault_recovery	bool	[ ]	y_6: fault_recovery
			self_good	bool	[ ]	y_1: self_good
		consequent	dtx_off	ltrap	[0.4, 0.6]	y_5: dtx_active
			dtx_on	rtrap	[0.4, 0.6]	y_5: dtx_active
y_6	$s_{6,1}$	antecedent	batt_critical	ltrap	[20, 30]	x_1: stored_energy
			faulty	bool	[ ]	x_{11}: fault_detected
			fault_recovery	bool	[ ]	y_6: fault_recovery
		consequent	fault_recovery_off	ltrap	[0.45, 0.55]	y_6: fault_recovery
			fault_recovery_on	rtrap	[0.45, 0.55]	y_6: fault_recovery

Table A.3 HEADS example - manually-tuned rule updates

y_j	$s_{j,p}$	$r_{j,p,w}$	Baseline/ Tuned	Rule
y_5	$s_{5,1}$	$r_{5,1,02}$	Baseline	batt_bad AND ((dt_sun_short OR sunlit) AND (COM_neighbors OR PROC_neighbors)) THEN dtx_on
			Tuned (Manual)	*batt_good* AND ((dt_sun_short OR sunlit) AND (COM_neighbors OR PROC_neighbors)) THEN dtx_on
	$s_{5,2}$	$r_{5,2,02}$	Baseline	NOT faulty AND NOT fault_recovery THEN dtx_on
			Tuned (Manual)	NOT faulty AND NOT fault_recovery *AND self_good* THEN dtx_on

Table A.4 HEADS example - manually-tuned parameter updates

y_j	$s_{j,p}$	Antecedent/ Consequent	Fuzzy Set	Activation Function	Baseline Parameters	Tuned Parameters
y_1	$s_{1,1}$	antecedent	batt_critical	ltrap	[10, 20]	[20, 30]
			batt_low	trap	[15, 20, 25, 30]	[15, 25, 35, 60]
			batt_good	trap	[30, 50, 70, 80]	[50, 60, 70, 90]
			batt_full	rtrap	[60, 80]	[80, 90]
		consequent	self_bad	ltrap	[0.4, 0.5]	[0.4, 0.45]
			self_marginal	trap	[0.45, 0.5, 0.55, 0.65]	[0.4, 0.5, 0.55, 0.65]
y_3	$s_{3,3}$	consequent	sensor_on	rtrap	[0.35, 0.45]	[0.35, 0.95]
	$s_{3,4}$	consequent	sensor_on	rtrap	[0.35, 0.7]	[0.35, 1.7]
y_4	$s_{4,1}$	consequent	sensor_off	ltrap	[0.5, 0.6]	[0.3, 0.6]
			sensor_on	rtrap	[0.5, 0.6]	[0.3, 0.6]
	$s_{4,2}$	consequent	sensor_off	ltrap	[0.3, 0.4]	[0.3, 0.55]
			sensor_on	rtrap	[0.6, 0.7]	[0.45, 0.7]
	$s_{4,3}$	consequent	sensor_off	ltrap	[0.3, 0.4]	[0.4, 0.5]
			sensor_on	rtrap	[0.35, 0.45]	[0.45, 1.55]
	$s_{4,4}$	consequent	sensor_off	ltrap	[0.3, 0.4]	[0.4, 0.5]
y_5	$s_{5,1}$	antecedent	batt_bad	ltrap	[30, 60]	[40, 50]
			batt_good	rtrap	[30, 30]	[40, 80]

Table A.5 HEADS example - genetic tuning parameter updates

y_j	$s_{j,p}$	Antecedent/ Consequent	Fuzzy Set	Activ. Func.	Manual Parameters	Genetic Parameters
y_1	$s_{1,1}$	antecedent	batt_critical	ltrap	[20, 30]	[22.21, 26.4]
			batt_low	trap	[15, 25, 35, 60]	[13.4, 31.06, 32.27, 38.52]
			batt_good	trap	[50, 60, 70, 90]	[48.49, 68.63, 81.54, 98.51]
			batt_full	rtrap	[80, 90]	[73.73, 81.28]
			dt_sun_short	ltrap	[300.0, 600.0]	[310.71, 467.92]

Table continues on next page

Table A.5 (continued)

y_j	$s_{j,p}$	Antecedent/ Consequent	Fuzzy Set	Activ. Func.	Manual Parameters	Genetic Parameters
			dt_sun_long	rtrap	[300.0, 900.0]	[349.63, 861.99]
			sunlit	bool	[ ]	[ ]
		consequent	self_bad	ltrap	[0.4, 0.45]	[0.22, 0.55]
			self_marginal	trap	[0.4, 0.5, 0.55, 0.65]	[0.42, 0.51, 0.54, 0.82]
			self_good	rtrap	[0.6, 0.7]	[0.63, 0.95]
	$s_{1,2}$	antecedent	batt_low	ltrap	[40, 60]	[27.19, 64.25]
			batt_good	rtrap	[50, 70]	[54.21, 65.95]
			faulty	bool	[ ]	[ ]
			fault_recovery	bool	[ ]	[ ]
		consequent	self_bad	ltrap	[0.3, 0.4]	[0.34, 0.42]
			self_marginal	trap	[0.45, 0.5, 0.55, 0.6]	[0.38, 0.8, 0.97, 0.99]
			self_good	rtrap	[0.55, 0.7]	[0.83, 0.88]
y_2	$s_{2,1}$	antecedent	batt_low	ltrap	[30, 40]	[25.11, 36.9]
			batt_good	tri	[30, 60]	[28.91, 47.84]
			batt_full	rtrap	[50, 60]	[36.95, 68.94]
			dt_sun_short	ltrap	[180.0, 300.0]	[236.24, 246.33]
			dt_sun_long	rtrap	[240.0, 480.0]	[192.59, 427.14]
			sunlit	bool	[ ]	[ ]
		consequent	sensor_off	ltrap	[0.4, 0.55]	[0.45, 0.52]
			sensor_on	rtrap	[0.45, 0.6]	[0.4, 0.89]
	$s_{2,2}$	antecedent	dt_target_short	ltrap	[300.0, 600.0]	[336.59, 557.68]
			dt_target_long	rtrap	[480.0, 780.0]	[484.2, 771.21]
			sensor_1_active	bool	[ ]	[ ]
			sensor_2_active	bool	[ ]	[ ]
			sensor_3_active	bool	[ ]	[ ]
			target_visible	bool	[ ]	[ ]

Table continues on next page

Table A.5 (continued)

y_j	$s_{j,p}$	Antecedent/ Consequent	Fuzzy Set	Activ. Func.	Manual Parameters	Genetic Parameters
		consequent	sensor_off	ltrap	[0.4, 0.55]	[0.4, 0.65]
			sensor_on	rtrap	[0.45, 0.6]	[0.46, 0.87]
	$s_{2,3}$	antecedent	sn_count_low	ltrap	[5, 7]	[3.99, 8.16]
			sn_count_high	rtrap	[6, 8]	[7.84, 8.66]
			sn_good_low	ltrap	[2, 3]	[2.25, 3.21]
			sn_good_high	rtrap	[2, 3]	[2.01, 2.2]
			sn_bad_low	ltrap	[3, 4]	[2.94, 3.04]
			sn_bad_high	rtrap	[3, 4]	[2.94, 56.2]
		consequent	sensor_off	ltrap	[0.2, 0.4]	[0.27, 0.35]
			sensor_on	rtrap	[0.3, 0.5]	[0.32, 0.57]
	$s_{2,4}$	antecedent	faulty	bool	[ ]	[ ]
			self_good	bool	[ ]	[ ]
			fault_recovery	bool	[ ]	[ ]
		consequent	sensor_off	ltrap	[0.5, 0.6]	[0.37, 0.49]
			sensor_on	rtrap	[0.5, 0.6]	[0.51, 0.66]
y_3	$s_{3,1}$	antecedent	batt_low	ltrap	[40, 60]	[40.05, 58.31]
			batt_good	tri	[55, 65]	[58.61, 79.72]
			batt_full	rtrap	[60, 70]	[65.78, 76.79]
			dt_sun_short	ltrap	[60.0, 180.0]	[69.33, 223.55]
			dt_sun_long	rtrap	[120.0, 240.0]	[114.34, 253.39]
			sunlit	bool	[ ]	[ ]
		consequent	sensor_off	ltrap	[0.5, 0.6]	[0.47, 0.57]
			sensor_on	rtrap	[0.5, 0.6]	[0.3, 0.6]
	$s_{3,2}$	antecedent	dt_target_short	ltrap	[60.0, 180.0]	[60.23, 159.37]
			dt_target_long	rtrap	[120.0, 300.0]	[131.68, 242.66]
			sensor_1_active	bool	[ ]	[ ]
			sensor_2_active	bool	[ ]	[ ]

Table continues on next page

Table A.5 (continued)

y_j	$s_{j,p}$	Antecedent/ Consequent	Fuzzy Set	Activ. Func.	Manual Parameters	Genetic Parameters
			sensor_3_active	bool	[ ]	[ ]
			target_visible	bool	[ ]	[ ]
		consequent	sensor_off	ltrap	[0.4, 0.6]	[0.43, 0.54]
			sensor_on	rtrap	[0.5, 0.6]	[0.46, 0.62]
	$s_{3,3}$	antecedent	sn_count_low	ltrap	[7, 8]	[5.95, 10.27]
			sn_count_high	rtrap	[7, 8]	[6.52, 8.99]
			sn_good_low	ltrap	[4, 5]	[3.76, 3.9]
			sn_good_high	rtrap	[4, 5]	[4.59, 6.32]
			sn_bad_low	ltrap	[1, 2]	[1.27, 1.35]
			sn_bad_high	rtrap	[1, 2]	[0.77, 2.66]
		consequent	sensor_off	ltrap	[0.3, 0.4]	[0.35, 0.42]
			sensor_on	rtrap	[0.35, 0.95]	[0.42, 0.84]
	$s_{3,4}$	antecedent	faulty	bool	[ ]	[ ]
			self_good	bool	[ ]	[ ]
			fault_recovery	bool	[ ]	[ ]
		consequent	sensor_off	ltrap	[0.3, 0.4]	[0.24, 0.43]
			sensor_on	rtrap	[0.35, 1.7]	[0.34, 0.48]
y_4	$s_{4,1}$	antecedent	batt_low	ltrap	[50, 60]	[53.66, 86.13]
			batt_good	tri	[55, 70]	[79.04, 79.76]
			batt_full	rtrap	[60, 80]	[60.21, 65.73]
			dt_sun_short	ltrap	[60.0, 120.0]	[56.26, 111.39]
			dt_sun_long	rtrap	[60.0, 120.0]	[66.6, 96.37]
			sunlit	bool	[ ]	[ ]
		consequent	sensor_off	ltrap	[0.3, 0.6]	[0.27, 0.57]
			sensor_on	rtrap	[0.3, 0.6]	[0.34, 0.67]
	$s_{4,2}$	antecedent	dt_target_short	ltrap	[60.0, 120.0]	[55.72, 125.28]
			dt_target_long	rtrap	[60.0, 120.0]	[66.18, 140.98]

Table continues on next page

Table A.5 (continued)

y_j	$s_{j,p}$	Antecedent/ Consequent	Fuzzy Set	Activ. Func.	Manual Parameters	Genetic Parameters
			sensor_1_active	bool	[ ]	[ ]
			sensor_2_active	bool	[ ]	[ ]
			sensor_3_active	bool	[ ]	[ ]
			target_visible	bool	[ ]	[ ]
		consequent	sensor_off	ltrap	[0.3, 0.55]	[0.29, 0.56]
			sensor_on	rtrap	[0.45, 0.7]	[0.46, 0.71]
	$s_{4,3}$	antecedent	sn_count_low	ltrap	[6, 8]	[7.19, 19.88]
			sn_count_high	rtrap	[7, 8]	[7.33, 9.5]
			sn_good_low	ltrap	[3, 5]	[3.84, 4.27]
			sn_good_high	rtrap	[4, 6]	[2.99, 5.51]
			sn_bad_low	ltrap	[1, 3]	[1.1, 2.96]
			sn_bad_high	rtrap	[2, 3]	[2.37, 2.65]
		consequent	sensor_off	ltrap	[0.4, 0.5]	[0.29, 0.42]
			sensor_on	rtrap	[0.45, 1.55]	[0.42, 0.47]
	$s_{4,4}$	antecedent	faulty	bool	[ ]	[ ]
			self_good	bool	[ ]	[ ]
			fault_recovery	bool	[ ]	[ ]
		consequent	sensor_off	ltrap	[0.4, 0.5]	[0.42, 0.51]
			sensor_on	rtrap	[0.5, 0.6]	[0.44, 0.44]
y_5	$s_{5,1}$	antecedent	batt_bad	ltrap	[40, 50]	[42.99, 55.09]
			batt_good	rtrap	[40, 80]	[35.18, 84.49]
			dt_sun_short	ltrap	[300.0, 540.0]	[237.88, 499.46]
			dt_sun_long	rtrap	[420.0, 600.0]	[398.78, 622.25]
			COM_neighbors	rtrap	[0, 1]	[0.0, 0.98]
			PROC_neighbors	rtrap	[0, 1]	[0.0, 1.29]
			sensor_1_active	bool	[ ]	[ ]
			sensor_2_active	bool	[ ]	[ ]

Table continues on next page

Table A.5 (continued)

y_j	$s_{j,p}$	Antecedent/ Consequent	Fuzzy Set	Activ. Func.	Manual Parameters	Genetic Parameters
			sensor_3_active	bool	[ ]	[ ]
			sunlit	bool	[ ]	[ ]
		consequent	dtx_off	ltrap	[0.4, 0.6]	[0.45, 0.6]
			dtx_on	rtrap	[0.4, 0.6]	[0.38, 0.65]
	$s_{5,2}$	antecedent	faulty	bool	[ ]	[ ]
			self_good	bool	[ ]	[ ]
			fault_recovery	bool	[ ]	[ ]
		consequent	dtx_off	ltrap	[0.4, 0.6]	[0.37, 0.69]
			dtx_on	rtrap	[0.4, 0.6]	[0.4, 0.48]
y_6	$s_{6,1}$	antecedent	batt_critical	ltrap	[20, 30]	[18.52, 29.6]
			faulty	bool	[ ]	[ ]
			fault_recovery	bool	[ ]	[ ]
		consequent	fault_recovery_off	ltrap	[0.45, 0.55]	[0.43, 0.82]
			fault_recovery_on	rtrap	[0.45, 0.55]	[0.48, 0.64]

Appendix B: Example HEADS Complexity Metrics

Table B.1 shows the rules, fuzzy sets, and rule compactness complexity metrics.

Tables B.2-B.10 provide the rule similarity matrices for each expert. Note that the ensembles for the sensor activation outputs (y_2, y_3, and y_4) share a structure and differ only in fuzzy set parameterization. This leads the experts in these ensembles to share rule similarity matrices.

Table B.11 shows the number of fuzzy sets using each signal at each level in the hierarchy.

Table B.1 Example HEADS complexity metrics for rules and fuzzy sets

Hierarchy	ψ_{rules}	ψ_{fsets}	$\psi_{w,ANT}$	ψ_{ANT}	$\psi_{\overline{ANT}}$
System	74	126		5	1.7
y_1	10	17		3	1.75
s_1_1	6	10		3	2.0
r_1_1_01			1		
r_1_1_02			2		
r_1_1_03			3		
r_1_1_04			2		
r_1_1_05			3		
r_1_1_06			1		
s_1_2	4	7		2	1.5
r_1_2_01			2		
r_1_2_02			2		
r_1_2_03			1		
r_1_2_04			1		

Table continues on next page

Table B.1 (continued)

Hierarchy	ψ_{rules}	ψ_{fsets}	$\psi_{w,ANT}$	ψ_{ANT}	$\psi_{\overline{ANT}}$
y_2	17	29		3	1.475
s_2_1	4	8		3	2.0
r_2_1_01			1		
r_2_1_02			3		
r_2_1_03			3		
r_2_1_04			1		
s_2_2	5	8		3	1.4
r_2_2_01			1		
r_2_2_02			1		
r_2_2_03			1		
r_2_2_04			1		
r_2_2_05			3		
s_2_3	4	8		2	1.5
r_2_3_01			1		
r_2_3_02			1		
r_2_3_03			2		
r_2_3_04			2		
s_2_4	4	5		1	1.0
r_2_4_01			1		
r_2_4_02			1		
r_2_4_03			1		
r_2_4_04			1		
y_3	17	29		3	1.475
s_3_1	4	8		3	2.0
r_3_1_01			1		
r_3_1_02			3		
r_3_1_03			3		
r_3_1_04			1		
s_3_2	5	8		3	1.4

Table continues on next page

Table B.1 (continued)

Hierarchy	ψ_{rules}	ψ_{fsets}	$\psi_{w,ANT}$	ψ_{ANT}	$\psi_{\overline{ANT}}$
r_3_2_01			1		
r_3_2_02			1		
r_3_2_03			1		
r_3_2_04			1		
r_3_2_05			3		
s_3_3	4	8		2	1.5
r_3_3_01			1		
r_3_3_02			1		
r_3_3_03			2		
r_3_3_04			2		
s_3_4	4	5		1	1.0
r_3_4_01			1		
r_3_4_02			1		
r_3_4_03			1		
r_3_4_04			1		
y_4	17	29		3	1.475
s_4_1	4	8		3	2.0
r_4_1_01			1		
r_4_1_02			3		
r_4_1_03			3		
r_4_1_04			1		
s_4_2	5	8		3	1.4
r_4_2_01			1		
r_4_2_02			1		
r_4_2_03			1		
r_4_2_04			1		
r_4_2_05			3		
s_4_3	4	8		2	1.5
r_4_3_01			1		

Table continues on next page

Table B.1 (continued)

Hierarchy	ψ_{rules}	ψ_{fsets}	$\psi_{w,ANT}$	ψ_{ANT}	$\psi_{\overline{ANT}}$
r_4_3_02			1		
r_4_3_03			2		
r_4_3_04			2		
s_4_4	4	5		1	1.0
r_4_4_01			1		
r_4_4_02			1		
r_4_4_03			1		
r_4_4_04			1		
y_5	8	17		5	2.63
s_5_1	5	12		5	3.6
r_5_1_01			1		
r_5_1_02			5		
r_5_1_03			3		
r_5_1_04			4		
r_5_1_05			5		
s_5_2	3	5		2	1.67
r_5_2_01			2		
r_5_2_02			2		
r_5_2_03			1		
y_6	5	5		2	1.4
s_6_1	5	5		2	1.4
r_6_1_01			1		
r_6_1_02			1		
r_6_1_03			1		
r_6_1_04			2		
r_6_1_05			2		

Table B.2 Example HEADS rule similarity matrix for expert $s_{1,1}$

Rule	r_1_1_01	r_1_1_02	r_1_1_03	r_1_1_04	r_1_1_05	r_1_1_06
r_1_1_01	1.0	0.0	0.0	0.0	0.0	0.0
r_1_1_02	0.0	1.0	0.1	0.1	0.0	0.0
r_1_1_03	0.0	0.1	1.0	0.0	0.2	0.0
r_1_1_04	0.0	0.1	0.0	1.0	0.1	0.0
r_1_1_05	0.0	0.0	0.2	0.1	1.0	0.0
r_1_1_06	0.0	0.0	0.0	0.0	0.0	1.0

Table B.3 Example HEADS rule similarity matrix for expert $s_{1,2}$

Rule	r_1_2_01	r_1_2_02	r_1_2_03	r_1_2_04
r_1_2_01	1.0	0.14	0.14	0.0
r_1_2_02	0.14	1.0	0.14	0.0
r_1_2_03	0.14	0.14	1.0	0.0
r_1_2_04	0.0	0.0	0.0	1.0

Table B.4 Example HEADS rule similarity matrix for expert $s_{2,1}$, $s_{3,1}$, and $s_{4,1}$

Rule	r_j_1_01	r_j_1_02	r_j_1_03	r_j_1_04
r_j_1_01	1.0	0.0	0.0	0.0
r_j_1_02	0.0	1.0	0.25	0.0
r_j_1_03	0.0	0.25	1.0	0.0
r_j_1_04	0.0	0.0	0.0	1.0

Table B.5 Example HEADS rule similarity matrix for expert $s_{2,2}$, $s_{3,2}$, and $s_{4,2}$

Rule	r_j_2_01	r_j_2_02	r_j_2_03	r_j_2_04	r_j_2_05
r_j_2_01	1.0	0.0	0.0	0.125	0.0
r_j_2_02	0.0	1.0	0.0	0.0	0.0
r_j_2_03	0.0	0.0	1.0	0.0	0.0
r_j_2_04	0.125	0.0	0.0	1.0	0.0
r_j_2_05	0.0	0.0	0.0	0.0	1.0

Table B.6 Example HEADS rule similarity matrix for expert $s_{2,3}$, $s_{3,3}$, and $s_{4,3}$

Rule	r_j_3_01	r_j_3_02	r_j_3_03	r_j_3_04
r_j_3_01	1.0	0.0	0.0	0.0
r_j_3_02	0.0	1.0	0.0	0.0
r_j_3_03	0.0	0.0	1.0	0.0
r_j_3_04	0.0	0.0	0.0	1.0

Table B.7 Example HEADS rule similarity matrix for expert $s_{2,4}$, $s_{3,4}$, and $s_{4,4}$

Rule	r_j_4_01	r_j_4_02	r_j_4_03	r_j_4_04
r_j_4_01	1.0	0.2	0.0	0.0
r_j_4_02	0.2	1.0	0.0	0.0
r_j_4_03	0.0	0.0	1.0	0.0
r_j_4_04	0.0	0.0	0.0	1.0

Table B.8 Example HEADS rule similarity matrix for expert $s_{5,1}$

Rule	r_5_1_01	r_5_1_02	r_5_1_03	r_5_1_04	r_5_1_05
r_5_1_01	1.0	0.083	0.0	0.0	0.0
r_5_1_02	0.083	1.0	0.167	0.167	0.167
r_5_1_03	0.0	0.167	1.0	0.25	0.167
r_5_1_04	0.0	0.167	0.25	1.0	0.167
r_5_1_05	0.0	0.167	0.167	0.167	1.0

Table B.9 Example HEADS rule similarity matrix for expert $s_{5,2}$

Rule	r_5_2_01	r_5_2_02	r_5_2_03
r_5_2_01	1.0	0.4	0.0
r_5_2_02	0.4	1.0	0.0
r_5_2_03	0.0	0.0	1.0

Table B.10 Example HEADS rule similarity matrix for expert $s_{6,1}$

Rule	r_6_1_01	r_6_1_02	r_6_1_03	r_6_1_04	r_6_1_05
r_6_1_01	1.0	0.0	0.0	0.2	0.2
r_6_1_02	0.0	1.0	0.2	0.0	0.0
r_6_1_03	0.0	0.2	1.0	0.0	0.0
r_6_1_04	0.2	0.0	0.0	1.0	0.4
r_6_1_05	0.2	0.0	0.0	0.4	1.0

Table B.11 Example HEADS fuzzy set to signal mapping complexity

Hierarchy	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	y_1	y_2	y_3	y_4	y_5	y_6
System	18	5	10	3	6	1	1	6	6	6	6	4	4	4	4	0	6
y_1	6	1	2	0	0	0	0	0	0	0	1	0	0	0	0	0	1
s_1_1	4	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
s_1_2	2	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
y_2	3	1	2	1	2	0	0	2	2	2	1	1	1	1	1	0	1
s_2_1	3	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
s_2_2	0	0	0	1	2	0	0	0	0	0	0	0	1	1	1	0	0
s_2_3	0	0	0	0	0	0	0	2	2	2	0	0	0	0	0	0	0
s_2_4	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1
y_3	3	1	2	1	2	0	0	2	2	2	1	1	1	1	1	0	1
s_3_1	3	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
s_3_2	0	0	0	1	2	0	0	0	0	0	0	0	1	1	1	0	0
s_3_3	0	0	0	0	0	0	0	2	2	2	0	0	0	0	0	0	0
s_3_4	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1
y_4	3	1	2	1	2	0	0	2	2	2	1	1	1	1	1	0	1
s_4_1	3	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
s_4_2	0	0	0	1	2	0	0	0	0	0	0	0	1	1	1	0	0
s_4_3	0	0	0	0	0	0	0	2	2	2	0	0	0	0	0	0	0
s_4_4	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1
y_5	2	1	2	0	0	1	1	0	0	0	1	1	1	1	1	0	1
s_5_1	2	1	2	0	0	1	1	0	0	0	0	0	1	1	1	0	0
s_5_2	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1
y_6	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
s_6_1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1

Appendix C: HEADS Performance Scores

Table C.1 shows the performance for all nodes within each orbit as compared between baseline, manually-tuned, and evolutionarily-tuned systems. The 'minimum' metric is the lowest positive score of any node for the given scoring parameter. Total score sums across the other scoring metrics. Subtotal score is the sum of all sensor and high-bandwidth transmitter scoring. Self Good and Stored Energy scores are for individual signals of interest.

Table C.2 shows simulation, orbit, and per-satellite scores for the baseline (untuned) system design.

Table C.3 shows simulation, orbit, and per-satellite scores for the manually-tuned system design.

Table C.4 shows simulation, orbit, and per-satellite scores for the system design tuned via evolutionary algorithm.

Table C.1 HEADS system performance - baseline and tuned comparison

Collection	Score	Statistic	Baseline	Manual	Genetic
System	Total	Average	28686	42529	**45441**
		Maximum	51536	52474	**52992**
		Minimum	12845	12713	**12744**
	Subtotal	Average	14436	16925	**17194**
		Maximum	22756	23684	**24192**
		Minimum	**370**	238	269
	Self Good	Average	9940	13338	**13965**
		Maximum	**14400**	14400	14400
		Minimum	**10**	5	10

Table continues on next page

Table C.1 (continued)

Collection	Score	Statistic	Baseline	Manual	Genetic
	Stored Energy	Average	4309	12266	**14280**
		Maximum	**14400**	14400	14400
		Minimum	50	180	**12475**
SEN_A	Total	Average	31241	38863	**40415**
		Maximum	41285	42581	**43522**
		Minimum	**14923**	14853	14820
	Subtotal	Average	12105	11619	**12527**
		Maximum	**15982**	14364	14722
		Minimum	**973**	908	880
	Self Good	Average	12520	13420	**13528**
		Maximum	**14400**	14400	14400
		Minimum	**10**	5	10
	Stored Energy	Average	6614	13823	**14358**
		Maximum	**14400**	14400	14400
		Minimum	215	2035	**13805**
SEN_B	Total	Average	21000	24461	**36957**
		Maximum	29880	28670	**40928**
		Minimum	**12845**	12713	12744
	Subtotal	Average	**8939**	7965	8932
		Maximum	**12404**	10181	12128
		Minimum	**370**	238	269
	Self Good	Average	9390	11205	**13692**
		Maximum	**14400**	13210	14400
		Minimum	15	10	**7790**
	Stored Energy	Average	2670	5290	**14333**
		Maximum	**14400**	14400	14400
		Minimum	50	1850	**12475**
SEN_C	Total	Average	31678	47152	**49242**
		Maximum	50454	51369	**51536**

Table continues on next page

Table C.1 (continued)

Collection	Score	Statistic	Baseline	Manual	Genetic
		Minimum	25734	24665	**31274**
	Subtotal	Average	17246	20343	**20464**
		Maximum	21674	22579	**22746**
		Minimum	2377	2349	**2474**
	Self Good	Average	9485	13679	**14389**
		Maximum	**14400**	14400	14400
		Minimum	4970	4945	**14365**
	Stored Energy	Average	4946	13129	**14388**
		Maximum	**14400**	14400	14400
		Minimum	90	180	**14255**
SEN_D	Total	Average	28859	50352	**50664**
		Maximum	51536	52474	**52992**
		Minimum	24819	25763	**48238**
	Subtotal	Average	17065	**22678**	22302
		Maximum	22756	23684	**24192**
		Minimum	**20124**	18938	19608
	Self Good	Average	8621	14000	**14197**
		Maximum	**14400**	14400	14400
		Minimum	4695	5075	**14355**
	Stored Energy	Average	3172	13673	**14165**
		Maximum	**14400**	14400	14400
		Minimum	55	190	**13965**

Table C.2 HEADS system performance - baseline

Node/Collection	Total	Subtotal	Self Good	Stored Energy
Grand Total	5306937	2670762	1838960	797215
Grand Total (avg)	28686	14436	9940	4309
SEN_A	1562072	605287	626045	330740

Table continues on next page

Table C.2 (continued)

Node/Collection	Total	Subtotal	Self Good	Stored Energy
SEN_A (avg)	31241	12105	12520	6614
SEN_A (max)	41285	15982	14400	14400
SEN_A (min >0)	14923	973	10	215
SEN_B	630010	268190	281715	80105
SEN_B (avg)	21000	8939	9390	2670
SEN_B (max)	29880	12404	14400	14400
SEN_B (min >0)	12845	370	15	50
SEN_C	950357	517387	284565	148405
SEN_C (avg)	31678	17246	9485	4946
SEN_C (max)	50454	21674	14400	14400
SEN_C (min >0)	25734	2377	4970	90
SEN_D	2164498	1279898	646635	237965
SEN_D (avg)	28859	17065	8621	3172
SEN_D (max)	51536	22756	14400	14400
SEN_D (min >0)	24819	20124	4695	55
SEN_A0	17369	3554	10	13805
SEN_A1	39596	11571	14370	13655
SEN_A2	39491	11516	14220	13755
SEN_A3	28775	13655	13810	1310
SEN_A4	28896	13291	14390	1215
SEN_A5	41023	12228	14395	14400
SEN_A6	29703	1033	14270	14400
SEN_A7	28110	13005	14170	935
SEN_A8	41285	12485	14400	14400
SEN_A9	28951	13316	14375	1260
SEN_A10	25976	15556	8925	1495
SEN_A11	26478	15758	9010	1710
SEN_A12	26268	15708	8960	1600
SEN_A13	39964	11629	14385	13950

Table continues on next page

Table C.2 (continued)

Node/Collection	Total	Subtotal	Self Good	Stored Energy
SEN_A14	14923	973	10	13940
SEN_A15	40415	11715	14375	14325
SEN_A16	28763	13773	13620	1370
SEN_A17	28122	12982	14385	755
SEN_A18	27948	12958	14395	595
SEN_A19	28974	13324	14400	1250
SEN_A20	40193	11698	14380	14115
SEN_A21	40684	11884	14400	14400
SEN_A22	26586	15806	9005	1775
SEN_A23	29314	13559	14355	1400
SEN_A24	39540	11530	14375	13635
SEN_A25	26850	1015	11435	14400
SEN_A26	26280	15640	8970	1670
SEN_A27	39352	11517	14385	13450
SEN_A28	28468	13168	14375	925
SEN_A29	26857	15982	9070	1805
SEN_A30	28675	13230	14395	1050
SEN_A31	40884	12094	14390	14400
SEN_A32	25943	15548	8900	1495
SEN_A33	28158	13033	14385	740
SEN_A34	28886	13321	14380	1185
SEN_A35	40070	11650	14390	14030
SEN_A36	26377	15692	8950	1735
SEN_A37	27369	12619	14375	375
SEN_A38	41105	12325	14380	14400
SEN_A39	40534	11784	14395	14355
SEN_A40	39443	11478	14150	13815
SEN_A41	27876	12916	14390	570
SEN_A42	40920	12120	14400	14400

Table continues on next page

Table C.2 (continued)

Node/Collection	Total	Subtotal	Self Good	Stored Energy
SEN_A43	26928	12493	14220	215
SEN_A44	26429	15774	9000	1655
SEN_A45	27671	12851	14390	430
SEN_A46	24629	14444	9090	1095
SEN_A47	29857	1057	14400	14400
SEN_A48	26847	15972	9050	1825
SEN_A49	28317	13057	14390	870
SEN_B0	22130	9350	12160	620
SEN_B1	20157	12082	6740	1335
SEN_B2	20600	8455	12045	100
SEN_B3	20594	12404	6850	1340
SEN_B4	20103	12038	6720	1345
SEN_B5	16494	10884	4760	850
SEN_B6	20308	12248	6765	1295
SEN_B7	19660	11890	6665	1105
SEN_B8	29880	1080	14400	14400
SEN_B9	21682	8582	12165	935
SEN_B10	15400	985	15	14400
SEN_B11	22688	9623	12100	965
SEN_B12	21382	11012	9250	1120
SEN_B13	21818	10213	10650	955
SEN_B14	20579	9254	10990	335
SEN_B15	19951	11996	6695	1260
SEN_B16	21906	8581	12015	1310
SEN_B17	21873	9523	11550	800
SEN_B18	19650	11890	6660	1100
SEN_B19	21436	8466	12020	950
SEN_B20	22189	9424	12115	650
SEN_B21	21044	8544	12105	395

Table continues on next page

Table C.2 (continued)

Node/Collection	Total	Subtotal	Self Good	Stored Energy
SEN_B22	21938	8568	12155	1215
SEN_B23	12845	370	0	12475
SEN_B24	20313	12248	6770	1295
SEN_B25	29675	875	14400	14400
SEN_B26	20459	8339	12070	50
SEN_B27	20895	8530	12070	295
SEN_B28	20469	12199	6810	1460
SEN_B29	21892	8537	12005	1350
SEN_C0	26262	20732	5025	505
SEN_C1	25905	20510	4980	415
SEN_C2	30805	19525	11185	95
SEN_C3	31008	19513	11405	90
SEN_C4	0	0	0	0
SEN_C5	34186	19916	14065	205
SEN_C6	49522	20942	14395	14185
SEN_C7	31177	2377	14400	14400
SEN_C8	25734	20314	4970	450
SEN_C9	49495	20950	14330	14215
SEN_C10	30916	19651	11110	155
SEN_C11	25874	20434	4995	445
SEN_C12	49320	20520	14400	14400
SEN_C13	30883	19858	10775	250
SEN_C14	26054	20594	5025	435
SEN_C15	49970	21180	14390	14400
SEN_C16	33497	20132	13200	165
SEN_C17	0	0	0	0
SEN_C18	31499	19719	11655	125
SEN_C19	26155	20700	5055	400
SEN_C20	26326	20806	5065	455

Table continues on next page

Table C.2 (continued)

Node/Collection	Total	Subtotal	Self Good	Stored Energy
SEN_C21	0	0	0	0
SEN_C22	50454	21674	14380	14400
SEN_C23	26103	20708	5030	365
SEN_C24	49604	21074	14315	14215
SEN_C25	49852	21167	14390	14295
SEN_C26	32120	19725	12255	140
SEN_C27	26077	20682	4995	400
SEN_C28	31519	2719	14400	14400
SEN_C29	50040	21265	14375	14400
SEN_D0	0	0	0	0
SEN_D1	50415	21955	14320	14140
SEN_D2	50708	22048	14385	14275
SEN_D3	50647	22012	14375	14260
SEN_D4	26883	21388	5085	410
SEN_D5	0	0	0	0
SEN_D6	0	0	0	0
SEN_D7	50414	21924	14325	14165
SEN_D8	34999	20869	13880	250
SEN_D9	0	0	0	0
SEN_D10	26794	21324	5100	370
SEN_D11	26810	21350	5090	370
SEN_D12	26643	21258	5075	310
SEN_D13	0	0	0	0
SEN_D14	35005	21030	13735	240
SEN_D15	51122	22497	14375	14250
SEN_D16	31769	21124	10355	290
SEN_D17	0	0	0	0
SEN_D18	34846	20776	13910	160
SEN_D19	34942	20932	13835	175

Table continues on next page

Table C.2 (continued)

Node/Collection	Total	Subtotal	Self Good	Stored Energy
SEN_D20	50235	21915	14255	14065
SEN_D21	33870	21250	12510	110
SEN_D22	49832	21752	14175	13905
SEN_D23	34566	21271	13180	115
SEN_D24	33027	20667	12275	85
SEN_D25	26866	21416	5065	385
SEN_D26	26780	21370	5060	350
SEN_D27	50254	21944	14250	14060
SEN_D28	50087	21912	14220	13955
SEN_D29	0	0	0	0
SEN_D30	50133	21908	14185	14040
SEN_D31	32273	20973	11075	225
SEN_D32	30733	20448	10225	60
SEN_D33	32305	20970	11100	235
SEN_D34	31869	20674	11080	115
SEN_D35	0	0	0	0
SEN_D36	24819	20124	4695	0
SEN_D37	31884	20674	11105	105
SEN_D38	32240	20850	11200	190
SEN_D39	32258	20938	11070	250
SEN_D40	31402	20592	10730	80
SEN_D41	32322	20927	11130	265
SEN_D42	31627	20677	10850	100
SEN_D43	25070	20310	4760	0
SEN_D44	32116	21051	10740	325
SEN_D45	32130	21120	10660	350
SEN_D46	31911	20591	11265	55
SEN_D47	33907	21282	12230	395
SEN_D48	50751	21951	14400	14400

Table continues on next page

Table C.2 (continued)

Node/Collection	Total	Subtotal	Self Good	Stored Energy
SEN_D49	35038	20943	13845	250
SEN_D50	27542	21852	5200	490
SEN_D51	27269	21674	5160	435
SEN_D52	0	0	0	0
SEN_D53	35104	21089	13740	275
SEN_D54	51160	22380	14395	14385
SEN_D55	33076	20791	12145	140
SEN_D56	27430	21810	5165	455
SEN_D57	32482	20727	11630	125
SEN_D58	27384	21764	5165	455
SEN_D59	27453	21808	5190	455
SEN_D60	29586	21391	7810	385
SEN_D61	0	0	0	0
SEN_D62	0	0	0	0
SEN_D63	27059	21584	5085	390
SEN_D64	0	0	0	0
SEN_D65	51536	22756	14380	14400
SEN_D66	50904	22264	14385	14255
SEN_D67	0	0	0	0
SEN_D68	0	0	0	0
SEN_D69	50424	22004	14305	14115
SEN_D70	31214	20409	10700	105
SEN_D71	51390	22610	14380	14400
SEN_D72	26708	21308	5050	350
SEN_D73	34475	20690	13570	215
SEN_D74	0	0	0	0

Table C.3 HEADS system performance - manually-tuned

Node/Collection	Total	Subtotal	Self Good	Stored Energy
Grand Total	7868001	3131136	2467580	2269285
Grand Total (avg)	42529	16925	13338	12266
SEN_A	1943155	580975	671000	691180
SEN_A (avg)	38863	11619	13420	13823
SEN_A (max)	42581	14364	14400	14400
SEN_A (min >0)	14853	908	5	2035
SEN_B	733834	238974	336155	158705
SEN_B (avg)	24461	7965	11205	5290
SEN_B (max)	28670	10181	13210	14400
SEN_B (min >0)	12713	238	10	1850
SEN_C	1414583	610298	410390	393895
SEN_C (avg)	47152	20343	13679	13129
SEN_C (max)	51369	22579	14400	14400
SEN_C (min >0)	24665	2349	4945	180
SEN_D	3776429	1700889	1050035	1025505
SEN_D (avg)	50352	22678	14000	13673
SEN_D (max)	52474	23684	14400	14400
SEN_D (min >0)	25763	18938	5075	190
SEN_A0	17115	3300	10	13805
SEN_A1	39857	11597	14320	13940
SEN_A2	39489	11514	14220	13755
SEN_A3	41967	13167	14400	14400
SEN_A4	41752	12952	14400	14400
SEN_A5	41019	12224	14395	14400
SEN_A6	29818	1018	14400	14400
SEN_A7	41382	12597	14385	14400
SEN_A8	41283	12483	14400	14400
SEN_A9	41791	12991	14400	14400

Table continues on next page

Table C.3 (continued)

Node/Collection	Total	Subtotal	Self Good	Stored Energy
SEN_A10	42106	13306	14400	14400
SEN_A11	42339	13549	14390	14400
SEN_A12	42243	13448	14395	14400
SEN_A13	39960	11625	14385	13950
SEN_A14	14853	908	5	13940
SEN_A15	40411	11711	14375	14325
SEN_A16	41984	13184	14400	14400
SEN_A17	41811	13011	14400	14400
SEN_A18	41870	13080	14390	14400
SEN_A19	41797	12997	14400	14400
SEN_A20	40189	11694	14380	14115
SEN_A21	40680	11880	14400	14400
SEN_A22	42415	13635	14380	14400
SEN_A23	41993	13193	14400	14400
SEN_A24	39821	11561	14320	13940
SEN_A25	15349	949	0	14400
SEN_A26	42197	13397	14400	14400
SEN_A27	39776	11571	14295	13910
SEN_A28	41984	13184	14400	14400
SEN_A29	42565	13765	14400	14400
SEN_A30	41700	12900	14400	14400
SEN_A31	40878	12088	14390	14400
SEN_A32	42143	13348	14395	14400
SEN_A33	41799	12999	14400	14400
SEN_A34	41769	12969	14400	14400
SEN_A35	40066	11646	14390	14030
SEN_A36	42313	13528	14385	14400
SEN_A37	41561	12806	14385	14370
SEN_A38	41101	12321	14380	14400

Table continues on next page

Table C.3 (continued)

Node/Collection	Total	Subtotal	Self Good	Stored Energy
SEN_A39	40530	11780	14395	14355
SEN_A40	39439	11474	14150	13815
SEN_A41	41855	13070	14385	14400
SEN_A42	40910	12125	14385	14400
SEN_A43	41350	12565	14385	14400
SEN_A44	42318	13528	14390	14400
SEN_A45	29748	13958	12895	2895
SEN_A46	27469	14364	11070	2035
SEN_A47	29852	1052	14400	14400
SEN_A48	42581	13806	14375	14400
SEN_A49	41957	13157	14400	14400
SEN_B0	24958	8988	12350	3620
SEN_B1	27772	9867	12665	5240
SEN_B2	22061	8186	12025	1850
SEN_B3	27876	10181	12630	5065
SEN_B4	27932	9832	12680	5420
SEN_B5	26409	9394	12495	4520
SEN_B6	27624	10044	12590	4990
SEN_B7	27543	9603	12695	5245
SEN_B8	28670	1060	13210	14400
SEN_B9	23006	8301	12160	2545
SEN_B10	14940	525	15	14400
SEN_B11	26947	9247	12620	5080
SEN_B12	27210	9615	12645	4950
SEN_B13	26970	9405	12615	4950
SEN_B14	23942	8582	12215	3145
SEN_B15	27450	9755	12620	5075
SEN_B16	23339	8304	12200	2835
SEN_B17	26295	9095	12545	4655

Table continues on next page

Table C.3 (continued)

Node/Collection	Total	Subtotal	Self Good	Stored Energy
SEN_B18	27551	9651	12640	5260
SEN_B19	22978	8238	12170	2570
SEN_B20	24955	9055	12320	3580
SEN_B21	22498	8228	12105	2165
SEN_B22	23414	8304	12235	2875
SEN_B23	12713	238	0	12475
SEN_B24	27718	10043	12595	5080
SEN_B25	15170	760	10	14400
SEN_B26	22133	8058	12070	2005
SEN_B27	22430	8210	12075	2145
SEN_B28	28220	9955	12740	5525
SEN_B29	23110	8250	12220	2640
SEN_C0	51369	22579	14390	14400
SEN_C1	51184	22389	14395	14400
SEN_C2	50710	21935	14375	14400
SEN_C3	50739	21954	14385	14400
SEN_C4	24665	19540	4945	180
SEN_C5	49601	20801	14400	14400
SEN_C6	49855	21305	14345	14205
SEN_C7	30804	2349	14055	14400
SEN_C8	50905	22140	14365	14400
SEN_C9	49874	21329	14330	14215
SEN_C10	50155	21375	14380	14400
SEN_C11	50955	22180	14375	14400
SEN_C12	50361	21561	14400	14400
SEN_C13	50839	22049	14390	14400
SEN_C14	51105	22315	14390	14400
SEN_C15	50384	21594	14390	14400
SEN_C16	50872	22092	14380	14400

Table continues on next page

Table C.3 (continued)

Node/Collection	Total	Subtotal	Self Good	Stored Energy
SEN_C17	35011	18271	11520	5220
SEN_C18	50891	22111	14380	14400
SEN_C19	51123	22343	14380	14400
SEN_C20	51299	22519	14380	14400
SEN_C21	26414	19394	6680	340
SEN_C22	50669	21889	14380	14400
SEN_C23	51191	22406	14385	14400
SEN_C24	49949	21419	14315	14215
SEN_C25	50207	21522	14365	14320
SEN_C26	50920	22130	14390	14400
SEN_C27	51221	22461	14360	14400
SEN_C28	30881	2691	13790	14400
SEN_C29	50430	21655	14375	14400
SEN_D0	51563	22763	14400	14400
SEN_D1	50799	22314	14295	14190
SEN_D2	51144	22474	14380	14290
SEN_D3	51072	22437	14345	14290
SEN_D4	52101	23301	14400	14400
SEN_D5	52030	23235	14395	14400
SEN_D6	51734	22939	14395	14400
SEN_D7	50832	22312	14305	14215
SEN_D8	50589	21789	14400	14400
SEN_D9	50340	21540	14400	14400
SEN_D10	51943	23143	14400	14400
SEN_D11	51963	23178	14385	14400
SEN_D12	51816	23016	14400	14400
SEN_D13	26925	19970	6740	215
SEN_D14	51667	22867	14400	14400
SEN_D15	51365	22745	14335	14285

Table continues on next page

Table C.3 (continued)

Node/Collection	Total	Subtotal	Self Good	Stored Energy
SEN_D16	51805	23005	14400	14400
SEN_D17	25763	20498	5075	190
SEN_D18	51979	23189	14390	14400
SEN_D19	36358	18938	11675	5745
SEN_D20	50711	22261	14280	14170
SEN_D21	51870	23110	14375	14385
SEN_D22	50412	22082	14225	14105
SEN_D23	51887	23112	14375	14400
SEN_D24	51745	23055	14385	14305
SEN_D25	52062	23267	14395	14400
SEN_D26	51970	23190	14380	14400
SEN_D27	50812	22322	14280	14210
SEN_D28	50665	22235	14250	14180
SEN_D29	50211	21936	14150	14125
SEN_D30	50708	22248	14270	14190
SEN_D31	51645	22865	14380	14400
SEN_D32	50543	21958	14325	14260
SEN_D33	51657	22877	14380	14400
SEN_D34	51107	22337	14370	14400
SEN_D35	50447	22027	14240	14180
SEN_D36	50062	21672	14215	14175
SEN_D37	51012	22217	14395	14400
SEN_D38	51831	23041	14390	14400
SEN_D39	51669	22884	14385	14400
SEN_D40	50003	21213	14390	14400
SEN_D41	52037	23282	14355	14400
SEN_D42	50169	21389	14380	14400
SEN_D43	50550	22050	14265	14235
SEN_D44	51874	23099	14375	14400

Table continues on next page

Table C.3 (continued)

Node/Collection	Total	Subtotal	Self Good	Stored Energy
SEN_D45	51959	23174	14385	14400
SEN_D46	51129	22339	14390	14400
SEN_D47	52025	23240	14385	14400
SEN_D48	51667	22902	14365	14400
SEN_D49	51525	22730	14395	14400
SEN_D50	52456	23681	14375	14400
SEN_D51	52315	23520	14395	14400
SEN_D52	52369	23574	14395	14400
SEN_D53	52426	23636	14390	14400
SEN_D54	51567	22787	14395	14385
SEN_D55	52105	23330	14375	14400
SEN_D56	52474	23684	14390	14400
SEN_D57	52095	23305	14390	14400
SEN_D58	52464	23664	14400	14400
SEN_D59	52466	23671	14395	14400
SEN_D60	52234	23434	14400	14400
SEN_D61	52090	23305	14385	14400
SEN_D62	52160	23360	14400	14400
SEN_D63	52287	23502	14385	14400
SEN_D64	52328	23533	14395	14400
SEN_D65	51757	22977	14380	14400
SEN_D66	51312	22657	14385	14270
SEN_D67	27140	20050	6750	340
SEN_D68	52240	23440	14400	14400
SEN_D69	50832	22382	14280	14170
SEN_D70	51883	23103	14380	14400
SEN_D71	51636	22856	14380	14400
SEN_D72	52034	23239	14395	14400
SEN_D73	52052	23257	14395	14400

Table continues on next page

Table C.3 (continued)

Node/Collection	Total	Subtotal	Self Good	Stored Energy
SEN_D74	51955	23175	14380	14400

Table C.4 HEADS system performance - genetic tuning

Node/Collection	Total	Subtotal	Self Good	Stored Energy
Grand Total	8406604	3180969	2583700	2641935
Grand Total (avg)	45441	17194	13965	14280
SEN_A	2020750	626375	676445	717930
SEN_A (avg)	40415	12527	13528	14358
SEN_A (max)	43522	14722	14400	14400
SEN_A (min >0)	14820	880	10	13805
SEN_B	1108739	267989	410760	429990
SEN_B (avg)	36957	8932	13692	14333
SEN_B (max)	40928	12128	14400	14400
SEN_B (min >0)	12744	269	7790	12475
SEN_C	1477269	613939	431690	431640
SEN_C (avg)	49242	20464	14389	14388
SEN_C (max)	51536	22746	14400	14400
SEN_C (min >0)	31274	2474	14365	14255
SEN_D	3799846	1672666	1064805	1062375
SEN_D (avg)	50664	22302	14197	14165
SEN_D (max)	52992	24192	14400	14400
SEN_D (min >0)	48238	19608	14355	13965
SEN_A0	17130	3315	10	13805
SEN_A1	41537	12897	14380	14260
SEN_A2	41356	12816	14400	14140
SEN_A3	43046	14246	14400	14400
SEN_A4	43501	14701	14400	14400
SEN_A5	42249	13454	14395	14400

Table continues on next page

Table C.4 (continued)

Node/Collection	Total	Subtotal	Self Good	Stored Energy
SEN_A6	29854	1054	14400	14400
SEN_A7	42408	13623	14385	14400
SEN_A8	41991	13191	14400	14400
SEN_A9	43503	14703	14400	14400
SEN_A10	43006	14206	14400	14400
SEN_A11	43262	14462	14400	14400
SEN_A12	43174	14374	14400	14400
SEN_A13	41696	12946	14385	14365
SEN_A14	14820	880	0	13940
SEN_A15	41766	13001	14375	14390
SEN_A16	43046	14246	14400	14400
SEN_A17	42449	13649	14400	14400
SEN_A18	42396	13606	14390	14400
SEN_A19	43520	14720	14400	14400
SEN_A20	41785	13010	14380	14395
SEN_A21	41916	13116	14400	14400
SEN_A22	43340	14560	14380	14400
SEN_A23	43057	14257	14400	14400
SEN_A24	41508	12863	14375	14270
SEN_A25	15430	1030	0	14400
SEN_A26	43110	14310	14400	14400
SEN_A27	41561	12886	14385	14290
SEN_A28	42576	13776	14400	14400
SEN_A29	43465	14665	14400	14400
SEN_A30	42782	13982	14400	14400
SEN_A31	42123	13333	14390	14400
SEN_A32	43077	14277	14400	14400
SEN_A33	42456	13656	14400	14400
SEN_A34	43522	14722	14400	14400

Table continues on next page

Table C.4 (continued)

Node/Collection	Total	Subtotal	Self Good	Stored Energy
SEN_A35	41726	12966	14390	14370
SEN_A36	43209	14409	14400	14400
SEN_A37	42115	13330	14385	14400
SEN_A38	41911	13131	14380	14400
SEN_A39	41859	13064	14395	14400
SEN_A40	41250	12775	14370	14105
SEN_A41	42349	13564	14385	14400
SEN_A42	42156	13371	14385	14400
SEN_A43	41990	13205	14385	14400
SEN_A44	43273	14483	14390	14400
SEN_A45	42211	13421	14390	14400
SEN_A46	43437	14637	14400	14400
SEN_A47	29882	1082	14400	14400
SEN_A48	43483	14708	14375	14400
SEN_A49	42481	13696	14385	14400
SEN_B0	38721	9921	14400	14400
SEN_B1	39498	10718	14380	14400
SEN_B2	38050	9325	14385	14340
SEN_B3	39783	10993	14390	14400
SEN_B4	39477	10687	14390	14400
SEN_B5	39897	11097	14400	14400
SEN_B6	39628	10828	14400	14400
SEN_B7	39269	10469	14400	14400
SEN_B8	29908	1108	14400	14400
SEN_B9	38227	9432	14395	14400
SEN_B10	29490	690	14400	14400
SEN_B11	39913	11113	14400	14400
SEN_B12	40928	12128	14400	14400
SEN_B13	39097	10302	14395	14400

Table continues on next page

Table C.4 (continued)

Node/Collection	Total	Subtotal	Self Good	Stored Energy
SEN_B14	38401	9606	14395	14400
SEN_B15	39386	10606	14380	14400
SEN_B16	38252	9467	14385	14400
SEN_B17	38789	10004	14385	14400
SEN_B18	39341	10541	14400	14400
SEN_B19	38119	9329	14390	14400
SEN_B20	38791	9991	14400	14400
SEN_B21	38180	9390	14390	14400
SEN_B22	38251	9456	14395	14400
SEN_B23	12744	269	0	12475
SEN_B24	39596	10826	14370	14400
SEN_B25	22991	801	7790	14400
SEN_B26	37982	9202	14390	14390
SEN_B27	38155	9380	14390	14385
SEN_B28	39633	10858	14375	14400
SEN_B29	38242	9452	14390	14400
SEN_C0	51536	22746	14390	14400
SEN_C1	51369	22569	14400	14400
SEN_C2	50354	21579	14375	14400
SEN_C3	50366	21581	14385	14400
SEN_C4	50165	21365	14400	14400
SEN_C5	47377	18577	14400	14400
SEN_C6	49743	20983	14395	14365
SEN_C7	31274	2474	14400	14400
SEN_C8	51066	22301	14365	14400
SEN_C9	49633	20988	14390	14255
SEN_C10	50536	21756	14380	14400
SEN_C11	51131	22356	14375	14400
SEN_C12	50537	21737	14400	14400

Table continues on next page

Table C.4 (continued)

Node/Collection	Total	Subtotal	Self Good	Stored Energy
SEN_C13	50749	21959	14390	14400
SEN_C14	51280	22490	14390	14400
SEN_C15	49978	21188	14390	14400
SEN_C16	50573	21793	14380	14400
SEN_C17	50850	22050	14400	14400
SEN_C18	50581	21801	14380	14400
SEN_C19	51355	22555	14400	14400
SEN_C20	51499	22719	14380	14400
SEN_C21	49987	21187	14400	14400
SEN_C22	50489	21709	14380	14400
SEN_C23	51411	22611	14400	14400
SEN_C24	49789	21114	14390	14285
SEN_C25	49925	21200	14390	14335
SEN_C26	50599	21809	14390	14400
SEN_C27	51429	22629	14400	14400
SEN_C28	31632	2832	14400	14400
SEN_C29	50056	21281	14375	14400
SEN_D0	48364	19749	14400	14215
SEN_D1	50676	21936	14375	14365
SEN_D2	50854	22069	14385	14400
SEN_D3	50805	22030	14375	14400
SEN_D4	52270	23470	14400	14400
SEN_D5	51530	22735	14395	14400
SEN_D6	51227	22432	14395	14400
SEN_D7	50662	21947	14390	14325
SEN_D8	48267	19622	14400	14245
SEN_D9	48238	19608	14400	14230
SEN_D10	52165	23365	14400	14400
SEN_D11	52193	23393	14400	14400

Table continues on next page

Table C.4 (continued)

Node/Collection	Total	Subtotal	Self Good	Stored Energy
SEN_D12	52059	23259	14400	14400
SEN_D13	50759	21959	14400	14400
SEN_D14	48387	19722	14400	14265
SEN_D15	51185	22410	14375	14400
SEN_D16	51861	23061	14400	14400
SEN_D17	51497	22697	14400	14400
SEN_D18	51582	22792	14390	14400
SEN_D19	51726	22926	14400	14400
SEN_D20	50559	21969	14400	14190
SEN_D21	51440	22665	14375	14400
SEN_D22	50277	21827	14385	14065
SEN_D23	51477	22702	14375	14400
SEN_D24	51390	22605	14385	14400
SEN_D25	52301	23501	14400	14400
SEN_D26	52223	23423	14400	14400
SEN_D27	50579	21969	14395	14215
SEN_D28	50480	21925	14380	14175
SEN_D29	0	0	0	0
SEN_D30	50504	21939	14395	14170
SEN_D31	51510	22730	14380	14400
SEN_D32	50758	22043	14390	14325
SEN_D33	51482	22702	14380	14400
SEN_D34	51548	22778	14370	14400
SEN_D35	51125	22775	14385	13965
SEN_D36	50573	22098	14385	14090
SEN_D37	51577	22782	14395	14400
SEN_D38	51845	23055	14390	14400
SEN_D39	51925	23140	14385	14400
SEN_D40	51544	22754	14390	14400

Table continues on next page

Table C.4 (continued)

Node/Collection	Total	Subtotal	Self Good	Stored Energy
SEN_D41	51902	23147	14355	14400
SEN_D42	51632	22852	14380	14400
SEN_D43	50937	22342	14380	14215
SEN_D44	51622	22847	14375	14400
SEN_D45	51683	22898	14385	14400
SEN_D46	51363	22573	14390	14400
SEN_D47	51741	22956	14385	14400
SEN_D48	51638	22873	14365	14400
SEN_D49	51882	23087	14395	14400
SEN_D50	52589	23814	14375	14400
SEN_D51	52417	23617	14400	14400
SEN_D52	52992	24192	14400	14400
SEN_D53	51974	23184	14390	14400
SEN_D54	51114	22329	14395	14390
SEN_D55	51646	22871	14375	14400
SEN_D56	52571	23781	14390	14400
SEN_D57	51628	22838	14390	14400
SEN_D58	52589	23789	14400	14400
SEN_D59	52578	23783	14395	14400
SEN_D60	52311	23511	14400	14400
SEN_D61	51605	22820	14385	14400
SEN_D62	52797	24037	14400	14360
SEN_D63	52375	23575	14400	14400
SEN_D64	51828	23033	14395	14400
SEN_D65	51412	22632	14380	14400
SEN_D66	51003	22243	14385	14375
SEN_D67	48482	19812	14400	14270
SEN_D68	51704	22904	14400	14400
SEN_D69	50689	21979	14385	14325

Table continues on next page

Table C.4 (continued)

Node/Collection	Total	Subtotal	Self Good	Stored Energy
SEN_D70	51350	22570	14380	14400
SEN_D71	51274	22494	14380	14400
SEN_D72	52153	23353	14400	14400
SEN_D73	51534	22739	14395	14400
SEN_D74	51407	22627	14380	14400

About the Author

Peter Jorgensen is an advocate for curiosity, cross-disciplinary engineering, and human-centered design. He earned his B.S. (2013) and M.S. (2015) degrees in Mechanical Engineering from Marquette University with particular interest in electromechanical systems, dynamics, guidance, navigation, and controls (GN&C), and estimation theory, and pursued a Ph.D. (2022) in Electrical Engineering to substantiate his cross-disciplinary training and focus on controls theory-inspired solutions to enabling complex systems with on-board autonomy. While completing these degrees, he experimented with Computer Engineering, probability theory, Windows and Linux system administration, web application development, leadership, entrepreneurship, road and mountain biking, singing in the University Chorus, volunteering, fishing, automobile maintenance, computer gaming, and designing and building spacecraft. Peter mentors high school students in pre-engineering programs and undergraduate and graduate students who are passionate about exploring space technology through research and experiential learning.