NAVAL
POSTGRADUATE
SCHOOL

MONTEREY, CALIFORNIA

DISSERTATION

LEARNING FROM NOISY AND DELAYED REWARDS: THE VALUE OF REINFORCEMENT LEARNING TO DEFENSE MODELING AND SIMULATION

by

Jonathan Alt

September 2012

Dissertation Supervisor: Christian J. Darken

This thesis was performed at the MOVES Institute.
Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 2012	Dissertation

4. TITLE AND SUBTITLE: Learning from Noisy and Delayed Rewards: The Value of Reinforcement Learning to Defense Modeling and Simulation	5. FUNDING NUMBERS NA
6. AUTHOR(S): Jonathan K. Alt	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES: The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol Number: NA

12a. DISTRIBUTION / AVAILABILITY STATEMENT	12b. DISTRIBUTION CODE
Approved for public release; distribution is unlimited	A

13. ABSTRACT (maximum 200 words)

Modeling and simulation of military operations requires human behavior models capable of learning from experience in complex environments where feedback on action quality is noisy and delayed. This research examines the potential of reinforcement learning, a class of AI learning algorithms, to address this need. A novel reinforcement learning algorithm that uses the exponentially weighted average reward as an action-value estimator is described. Empirical results indicate that this relatively straight-forward approach improves learning speed in both benchmark environments and in challenging applied settings. Applications of reinforcement learning in the verification of the reward structure of a training simulation, the improvement in the performance of a discrete event simulation scheduling tool, and in enabling adaptive decision-making in combat simulation are presented. To place reinforcement learning within the context of broader models of human information processing, a practical cognitive architecture is developed and applied to the representation of a population within a conflict area. These varied applications and domains demonstrate that the potential for the use of reinforcement learning within modeling and simulation is great.

14. SUBJECT TERMS	15. NUMBER OF PAGES
reinforcement learning, architecture, agents, autonomous systems	321
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UU

Approved for public release; distribution is unlimited

LEARNING FROM NOISY AND DELAYED REWARDS: THE VALUE OF REINFORCEMENT LEARNING TO DEFENSE MODELING AND SIMULATION

Jonathan K. Alt
Lieutenant Colonel, United States Army
B.S., United States Military Academy, 1993
M.Ed., University of Georgia, 2001
M.S., Naval Postgraduate School, 2006

Submitted in partial fulfillment of the
requirements for the degree of

**DOCTOR OF PHILOSOPHY IN
MODELING, VIRTUAL ENVIRONMENTS, AND SIMULATION (MOVES)**

from the

**NAVAL POSTGRADUATE SCHOOL
September 2012**

Author:

_____
Jonathan Keith Alt

Approved By:

_____
Christian J. Darken, PhD
Associate Professor
Department of Computer Science
Dissertation Supervisor

_____
Michael M. McCauley, PhD
Research Professor
Department of Operations Research

_____
Michael Jaye, PhD
Associate Professor
Department of Defense Analysis

_____
Arnold Buss, PhD
Research Assistance Professor
Modeling Virtual Environments and Simulation

_____
Jeffrey Appleget, PhD
Senior Lecturer
Department of Operations Research

Approved By:

_____
Peter Denning, PhD, Professor and Chair, Department of Computer Science

Approved By:

_____
Douglas Moses, Associate Provost for Academic Affairs

ABSTRACT

Modeling and simulation of military operations requires human behavior models capable of learning from experience in complex environments where feedback on action quality is noisy and delayed. This research examines the potential of reinforcement learning, a class of AI learning algorithms, to address this need. A novel reinforcement learning algorithm that uses the exponentially weighted average reward as an action-value estimator is described. Empirical results indicate that this relatively straight-forward approach improves learning speed in both benchmark environments and in challenging applied settings. Applications of reinforcement learning in the verification of the reward structure of a training simulation, the improvement in the performance of a discrete event simulation scheduling tool, and in enabling adaptive decision-making in combat simulation are presented. To place reinforcement learning within the context of broader models of human information processing, a practical cognitive architecture is developed and applied to the representation of a population within a conflict area. These varied applications and domains demonstrate that the potential for the use of reinforcement learning within modeling and simulation is great.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I. INTRODUCTION 1

A. MOTIVATING PROBLEM 1

B. DEFINITIONS AND TERMINOLOGY 3

C. CHALLENGES OF ADAPTIVE AGENT DECISION MAKING IN ANALYTIC AND TRAINING SIMULATIONS 12

D. DISSERTATION CONTRIBUTIONS AND ORGANIZATION 13

II. REINFORCEMENT LEARNING AND COGNITIVE ARCHITECTURES FOR AUTONOMOUS AGENT DECISION-MAKING 17

A. REINFORCEMENT LEARNING 18

1. Temporal Differencing, Monte-Carlo, and TD(λ) Methods . . 22

a. Q-learning and Q(λ) 24

b. SARSA and SARSA(λ) 26

2. Exporation and Exploitation 29

a. ϵ-Greedy 30

b. Boltzmann Exploration 32

c. Interval Estimation and Upper Confidence Bound . . . 33

d. Other Recent Approaches 33

3. Measures of Learning Performance 35

a. Convergence 35

b. Speed of Convergence 35

c. Regret 35

d. Approximation Error 36

e. Frequency of Optimal Action Selection 37

B. COGNITIVE ARCHITECTURES 37

1. Considerations in Cognitive Modeling 38

2. Recognition-Primed Decision Making and Goals, Operators, Methods, Selectors 43

3. Review of Cognitive Architectures 45

a. Active Components of Thought 45

b. State Operator and Results 47

c. Connectionist Learning with Adaptive Rule Induction On-line 49

4. Learning in Cognitive Architectures 51

5. Agent Based Models, Cognitive Modeling, and Complexity . . 52

C. A SAMPLE OF DEPARTMENT OF DEFENSE SIMULATION MODELS . . . 64
1. Combined Arms Analysis Toolkit for the Twenty-first Century 64
2. UrbanSim . . . 65
3. Assignment Scheduling Capability for Unmanned Aerial Systems . . . 66
4. Cultural Geography Model . . . 66
D. SUMMARY . . . 68

III. DIRECT-Q COMPUTATION . . . 69
A. DIRECT-Q COMPUTATION . . . 69
B. BENCHMARK PROBLEMS . . . 76
1. N-Arm Bandit . . . 77
2. Two-Arm Bandit . . . 79
3. Gridworld . . . 81
4. Summary . . . 87
C. PHYSICAL TRAVELING SALESMAN PROBLEM . . . 88
1. Problem Specification . . . 89
2. RL Formulation . . . 92
3. Empirical Results . . . 93
4. Insights from the Physical Traveling Salesman Problem . . . 95
D. PACMAN . . . 96
1. Problem Specification . . . 97
2. RL Formulation . . . 97
3. Empirical Results . . . 99
4. Insights from PACMAN . . . 100
E. INSIGHTS ON DQ-C . . . 101

IV. APPLICATIONS . . . 103
A. UNMANNED AERIAL VEHICLE ASSIGNMENT AND SCHEDULING PROBLEM . . . 103
1. Assignment Scheduling Capability for UAVs . . . 104
2. Problem Specification . . . 104
3. RL Formulation and Empirical Results . . . 104
a. Case 1 Formulation . . . 105
b. Case 1 Empirical Results . . . 105
c. Case 2 Formulation . . . 107
d. Case 2 Empirical Results . . . 108
4. Insights on the Use of Reinforcement Learning in a Scheduling Tool . . . 111

B. ADAPTIVE BEHAVIOR IN COMBATXXI . . . 112
1. Case 1 Formulation . . . 114
2. Case 1 Empirical Results . . . 115
3. Case 2 Formulation . . . 116
4. Case 2 Empirical Results . . . 118
5. Insights on the Use of Reinforcement Learning in Combat Simulations . . . 119
C. VERIFYING THE REWARD STRUCTURE IN TRAINING SIMULATION . . . 120
1. Problem Specification . . . 121
2. RL Formulation and Empirical Results . . . 122
a. Case 1 Formulation . . . 122
b. Case 1 Empirical Results . . . 123
c. Case 2 Formulation . . . 129
d. Case 2 Empirical Results . . . 130
e. Case 3 Formulation . . . 134
f. Case 3 Empirical Results . . . 135
3. Insights on the Verification of Training Simulations . . . 142
D. CONCLUSIONS . . . 143

V. DEVELOPMENT OF A PRACTICAL COGNITIVE ARCHITECTURE 145
A. GENERAL FRAMEWORK AND IMPLEMENTATION DESCRIPTION . . . 145
1. Perception . . . 147
2. Meta-cognition . . . 148
3. Long-term Memory . . . 148
4. Action Selection . . . 149
B. GENERIC IMPLEMENTATION DESCRIPTION . . . 149
C. COGNITIVE ARCHITECTURES TO REPRESENT GROUP COGNITION . . . 154
1. Cognitive Structures . . . 155
2. Cognitive Processes . . . 157
3. Sentiment . . . 158
4. Group Behavior and Deviance . . . 159
D. APPLICATION OF THE REINFORCEMENT LEARNING AND A PRACTICAL COGNITIVE ARCHITECTURE WITHIN THE CULTURAL GEOGRAPHY MODEL . . . 160
1. Initial Application of Reinforcement Learning within CG . . . 161
2. Representing Theory of Planned Behavior . . . 163
3. Incorporation of a Cognitive Architecture . . . 166

E. CONCLUSIONS 167

VI. CONCLUSION 169

A. SUMMARY OF CONTRIBUTIONS 169
1. Direct-Q Computation 169
2. Enabling Adaptive Behavior in a Combat Simulation 170
3. Maximizing the Value of a UAV Schedule from a DES 170
4. Verification of Reward Structure in Training Simulation 171
5. Practical Cognitive Architecture 171

B. FUTURE RESEARCH 172

Appendix A: First-Visit and Every-Visit Monte-Carlo 175

Appendix B: Comparison of ϵ-greedy and Boltzmann Exploration 177

Appendix C: Bayesian Optimal Policy for N-Arm Bandit 183

Appendix D: Analysis of DQ-C and TD(λ) 187

Appendix E: PTSP Maps 193

Appendix F: Full ASC-U Formulation 203

Appendix G: Learned State-Action Values for UrbanSim 209

Appendix H: Strategy Level Policies by Algorithm UrbanSim 265

Appendix I: Additional Benchmarking Results 269

LIST OF REFERENCES 283

INITIAL DISTRIBUTION LIST 293

LIST OF FIGURES

Figure 1. Agent environment interaction. 3
Figure 2. Simple two state MDP. 4
Figure 3. Agent-environment interaction in reinforcement learning. 8
Figure 4. Information processing model of human cognition. 11
Figure 5. Information processing model of human cognition. 39
Figure 6. Recognition primed decision making 1231212312 44
Figure 7. ACT-R top-level conceptual diagram. 46
Figure 8. SOAR top-level conceptual diagram. 48
Figure 9. CLARION top-level conceptual diagram. 50
Figure 10. Cultural geography conceptual model. 67
Figure 11. Top three performers for 10-arm bandit benchmark domain, for $\sigma^2 = 1.0$ and $\sigma^2 = 0.0$, 250 trials and 500 replications. Mean total regret is plotted for each algorithm policy pair along with associated standard error. 78
Figure 12. Sample gridworld domain. 82
Figure 13. Top three performers for deterministic and stochastic 5x5 gridworld task, 500 trials and 1000 replications. Mean total utility for each is plotted for each algorithm policy pair along with associated standard error. 83
Figure 14. Top three performers for deterministic and stochastic 10x10 gridworld task, 500 trials and 1000 replications. Mean total utility for each is plotted for each algorithm policy pair along with associated standard error. 84
Figure 15. Top three performers for 5x5 gridworld, with oscillating transition matrix and a dynamic goal, 250 trials and 500 replications. Mean utility is plotted for each algorithm policy pair along with associated standard error. 87
Figure 16. Sample map for physical traveling salesman problem. 89
Figure 17. Mean waypoints obtained across all maps by top three performing algorithm and policy pairs. 94
Figure 18. Mean time per waypoints obtained across all maps by top three performing algorithm and policy pairs. 95
Figure 19. Medium sized classic Pacman map. 96

Figure 20. Mean percent of board cleared by all algorithm and policy pairs for best performing parameter settings. 100
Figure 21. Case 1 mission area timing. 106
Figure 22. Case 1 results. 107
Figure 23. Case 2 mission area timing. 109
Figure 24. Case 2 comparison mean value per iteration for each algorithm. 111
Figure 25. Typical modeling and simulation agent. 113
Figure 26. Modeling and simulation agent based on reinforcement learning. . . . 114
Figure 27. Route selection scenario COMBATXXI. 115
Figure 28. Route selection scenario COMBATXXI. 116
Figure 29. Formation selection scenario COMBATXXI. 117
Figure 30. Formation selection scenario COMBATXXI. 119
Figure 31. UrbanSim player interface. 121
Figure 32. Mean and standard error of the final score of a 15 turn game following 30 replications of each of the 27 strategy combinations. 124
Figure 33. Mean and standard error of the final score of a 15 turn game following 30 replications of each of the 162 strategy combinations. 125
Figure 34. Regret per turn and mean total regret for 162-arm bandit formulation of UrbanSim constant exploration rate. 127
Figure 35. Regret per turn and mean total regret for 162-arm bandit formulation of UrbanSim using decaying exploration rate. 128
Figure 36. Regret per 15 turn game and total regret over 1000 games with decisions made by agent by turn. 131
Figure 37. Mean score over 1000 games with decisions made by agent by turn for each algorithm policy pair. 132
Figure 38. End of game score for 1000 learning games and 300 greedy games. . . 136
Figure 39. Practical cognitive architecture full conceptual model. 146
Figure 40. top-level view of DES cognitive architecture. 150
Figure 41. Kenrick's updated to Maslow's hierarchy of needs. 153
Figure 42. Analysis methodology for close formed use case. 161
Figure 43. Population agent functional decomposition. 162
Figure 44. Distribution of threat actions. 163
Figure 45. Map 2 for physical traveling salesman problem. 193
Figure 46. Map 3 for physical traveling salesman problem. 194
Figure 47. Map 4 for physical traveling salesman problem. 195
Figure 48. Map 5 for physical traveling salesman problem. 196

Figure 49. Map 6 for physical traveling salesman problem. 197
Figure 50. Map 7 for physical traveling salesman problem. 198
Figure 51. Map 8 for physical traveling salesman problem. 199
Figure 52. Map 9 for physical traveling salesman problem. 200
Figure 53. Map 10 for physical traveling salesman problem. 201
Figure 54. Performance of DQ-C in 5x5 deterministic grid world over γ by α paired with ϵ-greedy. 269
Figure 55. Performance of DQ-C in 5x5 deterministic grid world over γ by α paired with Boltzmann exploration. 270
Figure 56. Performance of DQ-C in 5x5 deterministic grid world over γ by ϵ paired with ϵ-greedy with $\alpha = 0.6$. 271
Figure 57. Performance of DQ-C in 5x5 deterministic grid world over γ by τ paired with Boltzmann exploration with $\alpha = 0.6$. 272
Figure 58. Performance of SARSA(λ) in 5x5 deterministic grid world over γ by λ paired with ϵ-greedy. 273
Figure 59. Performance of SARSA(λ) in 5x5 deterministic grid world over γ by λ paired with Boltzmann exploration. 274
Figure 60. Performance of SARSA(λ) in 5x5 deterministic grid world over γ by α paired with ϵ-greedy with $\alpha = 0.6$. 275
Figure 61. Performance of SARSA(λ) in 5x5 deterministic grid world over γ by τ paired with Boltzmann exploration with $\alpha = 0.6$. 276
Figure 62. Performance of DQ-C in noisy 10-arm bandit over γ by α paired with ϵ-greedy. 277
Figure 63. Performance of DQ-C in noisy 10-arm bandit over γ by α paired with Botzmann exploration. 278
Figure 64. Performance of DQ-C in noisy 10-arm bandit over γ by τ paired with ϵ-greedy, $\alpha = 0.6$. 279
Figure 65. Performance of DQ-C in noisy 10-arm bandit over γ by τ paired with Botzmann exploration, $\alpha = 0.6$. 280

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

Table 1. Regret results for algorithms coupled with $epsilon-greedy$, β, for 10-arm bandit benchmark task following 250 trials with 500 replications with σ^2=0 on the reward signal. SARSA with ϵ-greedy provided best result. 79

Table 2. Regret results for algorithms coupled with $epsilon-greedy$, β, for 10-arm bandit benchmark task following 250 trials with 500 replications with σ^2=1 on the reward signal. DQ-C with β provided best results. 79

Table 3. Regret results for algorithms coupled with $epsilon-greedy$ and β for 2-arm bandit following 250 trials with 500 replications with $\sigma^2 = 0$, $\delta = \mu^* - \mu_1 = 0.8$ and 0.2. DQ-C with β provides best results in $\delta = 0.8$ case, Q-learning and SARSA paired with $\epsilon - greedy$ in the $\delta = 0.2$ case. 80

Table 4. Regret results for algorithms coupled with $epsilon - greedy$ and β for 2-arm bandit following 250 trials with 500 replications with $\sigma^2 = 1$, $\delta = \mu^* - \mu_1 = 0.8$ and 0.2. $Q(\lambda)$ paired with β resulted in the best performance for $\delta = 0.8$ and DQ-C paired with β for $\delta = 0.2$. 81

Table 5. Mean goals achieved for deterministic (top) and stochastic (bottom) 5x5 grid world following 500 trials with 1000 replications with single reward upon attainment of the goal state by algorithm and policy. DQ-C with ϵ-greedy obtained the best result in each case. 85

Table 6. Mean goals achieved for deterministic (top) and stochastic (bottom) 10x10 grid world following 500 trials with 500 replications with single reward upon attainment of the goal state by algorithm and policy. DQ-C with ϵ-greedy obtained the best result in each case. 86

Table 7. Scenario configuration parameters for ASC-U Case 2. 110

Table 8. Recommended strategies following 1000 training sessions by algorithm policy pair. 126

Table 9. Recommended strategies following 500 training sessions by algorithm policy pair using a decaying exploration strategy. 129

Table 10. Learned policy for Battalion Commander by turn for 15 turn UrbanSim game using DQ-C. 133

Table 11. Learned policy for all agents by turn for 15 turn UrbanSim game using DQ-C, ϵ-greedy. 134

Table 12. Learned policy for all agents by turn for 15 turn UrbanSim game using DQ-C, Boltzmann. 134

Table 13. Actions available by agent each turn . . . 135
Table 14. Learned action policy for all agents by turn for 15 turn UrbanSim game using DQ-C, ϵ-greedy. . . . 137
Table 15. Learned actions for all agents from turn 1-3 in 15 turn UrbanSim game using DQ-C, ϵ-greedy. . . . 138
Table 16. Learned actions for all agents from turn 4-6 in 15 turn UrbanSim game using DQ-C, ϵ-greedy. . . . 139
Table 17. Learned actions for all agents from turn 7-9 in 15 turn UrbanSim game using DQ-C, ϵ-greedy. . . . 140
Table 18. Learned actions for all agents from turn 10-12 in 15 turn UrbanSim game using DQ-C, ϵ-greedy. . . . 141
Table 19. Learned actions for all agents from turn 13-15 in 15 turn UrbanSim game using DQ-C, ϵ-greedy. . . . 142
Table 20. Learned policy for Battalion Commander by turn for 15 turn UrbanSim game using DQ-C. . . . 209
Table 21. Learned policy for CA unit by turn for 15 turn UrbanSim game using DQ-C. . . . 210
Table 22. Learned policy for E CO a by turn for 15 turn UrbanSim game using DQ-C. . . . 211
Table 23. Learned policy for F CO a by turn for 15 turn UrbanSim game using DQ-C. . . . 212
Table 24. Learned policy for E CO b by turn for 15 turn UrbanSim game using DQ-C. . . . 213
Table 25. Learned policy for F CO b by turn for 15 turn UrbanSim game using DQ-C. . . . 214
Table 26. Learned policy for G CO a by turn for 15 turn UrbanSim game using DQ-C. . . . 215
Table 27. Learned policy for H CO a by turn for 15 turn UrbanSim game using DQ-C. . . . 216
Table 28. Learned policy for G CO b by turn for 15 turn UrbanSim game using DQ-C. . . . 217
Table 29. Learned policy for QRF by turn for 15 turn UrbanSim game using DQ-C. . . . 218
Table 30. Learned policy for H CO b by turn for 15 turn UrbanSim game using DQ-C. . . . 219
Table 31. Learned policy for Battalion Commander by turn for 15 turn UrbanSim game using Q(λ). . . . 220

Table 32. Learned policy for CA unit by turn for 15 turn UrbanSim game using Q(λ). . . . 221
Table 33. Learned policy for E CO b by turn for 15 turn UrbanSim game using Q(λ). . . . 222
Table 34. Learned policy for E CO a by turn for 15 turn UrbanSim game using Q(λ). . . . 223
Table 35. Learned policy for F CO a by turn for 15 turn UrbanSim game using Q(λ). . . . 224
Table 36. Learned policy for F CO b by turn for 15 turn UrbanSim game using Q(λ). . . . 225
Table 37. Learned policy for G CO b by turn for 15 turn UrbanSim game using Q(λ). . . . 226
Table 38. Learned policy for G CO a by turn for 15 turn UrbanSim game using Q(λ). . . . 227
Table 39. Learned policy for H CO a by turn for 15 turn UrbanSim game using Q(λ). . . . 228
Table 40. Learned policy for QRF by turn for 15 turn UrbanSim game using Q(λ). 229
Table 41. Learned policy for H CO b by turn for 15 turn UrbanSim game using Q(λ). . . . 230
Table 42. Learned policy for Battalion Commander by turn for 15 turn UrbanSim game using SARSA(λ). . . . 231
Table 43. Learned policy for CA unit by turn for 15 turn UrbanSim game using SARSA(λ). . . . 232
Table 44. Learned policy for E CO a by turn for 15 turn UrbanSim game using SARSA(λ). . . . 233
Table 45. Learned policy for E CO b by turn for 15 turn UrbanSim game using SARSA(λ). . . . 234
Table 46. Learned policy for F CO b by turn for 15 turn UrbanSim game using SARSA(λ). . . . 235
Table 47. Learned policy for F CO a by turn for 15 turn UrbanSim game using SARSA(λ). . . . 236
Table 48. Learned policy for G CO a by turn for 15 turn UrbanSim game using SARSA(λ). . . . 237
Table 49. Learned policy for G CO b by turn for 15 turn UrbanSim game using SARSA(λ). . . . 238
Table 50. Learned policy for H CO a by turn for 15 turn UrbanSim game using SARSA(λ). . . . 239

Table 51. Learned policy for QRF by turn for 15 turn UrbanSim game using SARSA(λ). . . . 240
Table 52. Learned policy for H CO b by turn for 15 turn UrbanSim game using SARSA(λ). . . . 241
Table 53. Learned policy for Battalion Commander by turn for 15 turn UrbanSim game using DQ-C, Boltzmann. . . . 242
Table 54. Learned policy for CA unit by turn for 15 turn UrbanSim game using DQ-C, Boltzmann. . . . 243
Table 55. Learned policy for E CO a by turn for 15 turn UrbanSim game using DQ-C, Boltzmann. . . . 244
Table 56. Learned policy for E CO b by turn for 15 turn UrbanSim game using DQ-C, Boltzmann. . . . 245
Table 57. Learned policy for F CO a by turn for 15 turn UrbanSim game using DQ-C, Boltzmann. . . . 246
Table 58. Learned policy for F CO b by turn for 15 turn UrbanSim game using DQ-C, Boltzmann. . . . 247
Table 59. Learned policy for G CO a by turn for 15 turn UrbanSim game using DQ-C, Boltzmann. . . . 248
Table 60. Learned policy for G CO b by turn for 15 turn UrbanSim game using DQ-C, Boltzmann. . . . 249
Table 61. Learned policy for H CO a by turn for 15 turn UrbanSim game using DQ-C, Boltzmann. . . . 250
Table 62. Learned policy for H CO b by turn for 15 turn UrbanSim game using DQ-C, Boltzmann. . . . 251
Table 63. Learned policy for QRF by turn for 15 turn UrbanSim game using DQ-C, Boltzmann. . . . 252
Table 64. Learned policy for Battalion Commander by turn for 15 turn UrbanSim game using Q(λ), Boltzmann. . . . 253
Table 65. Learned policy for CA unit by turn for 15 turn UrbanSim game using Q(λ), Boltzmann. . . . 254
Table 66. Learned policy for E CO a by turn for 15 turn UrbanSim game using Q(λ), Boltzmann. . . . 255
Table 67. Learned policy for F CO a by turn for 15 turn UrbanSim game using Q(λ), Boltzmann. . . . 256
Table 68. Learned policy for E CO b by turn for 15 turn UrbanSim game using Q(λ), Boltzmann. . . . 257
Table 69. Learned policy for F CO b by turn for 15 turn UrbanSim game using Q(λ), Boltzmann. . . . 258

Table 70. Learned policy for G CO b by turn for 15 turn UrbanSim game using Q(λ), Boltzmann. . . . 259
Table 71. Learned policy for G CO a by turn for 15 turn UrbanSim game using Q(λ), Boltzmann. . . . 260
Table 72. Learned policy for H CO a by turn for 15 turn UrbanSim game using Q(λ), Boltzmann. . . . 261
Table 73. Learned policy for H CO b by turn for 15 turn UrbanSim game using Q(λ), Boltzmann. . . . 262
Table 74. Learned policy for QRF by turn for 15 turn UrbanSim game using Q(λ), Boltzmann. . . . 263
Table 75. Learned policy for all agents by turn for 15 turn UrbanSim game using Q(λ), ϵ-greedy. . . . 265
Table 76. Learned policy for all agents by turn for 15 turn UrbanSim game using DQ-C, ϵ-greedy. . . . 265
Table 77. Learned policy for all agents by turn for 15 turn UrbanSim game using DQ-C, Boltzmann. . . . 266
Table 78. Learned policy for all agents by turn for 15 turn UrbanSim game using SARSA(λ), ϵ-greedy. . . . 266
Table 79. Learned policy for all agents by turn for 15 turn UrbanSim game using SARSA(λ), Boltzmann. . . . 267
Table 80. Learned policy for all agents by turn for 15 turn UrbanSim game using Q(λ), Boltzmann. . . . 267
Table 81. Parameter settings that produced the best observed values for noisy 10-arm bandit. . . . 281
Table 82. Parameter settings that produced the best observed values for 5x5 Deterministic Gridworld. . . . 281
Table 83. Parameter settings that produced the best observed values for 5x5 Stochastic Gridworld. . . . 281
abbreviations.tex

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS

ACMAS Agent Centered Multi-Agent Systems

ACT-R Atomic Components of Thought

ANSF Afghan National Security Forces

AoA Analysis of Alternatives

ASC-U Assignment and Scheduling Tool for Unmanned Aerial Vehicles

BI Behavioral Intention

CG Cultural Geography

CLARION Connectionist Learning with Adaptive Rule Induction On-line

CLIPS C Language Integrated Production System

COMBATXXI Combined Arms Analysis Toolkit for the 21st Century

DDR Downside Deviation Ratio

DES Discrete Event Simulation

DQ-C Direct-Q Computation

DoD Department of Defense

DMSO Defense Modeling Simulation Office

GOMS Goals, Operators, Methods, and Selection

ICT Institute for Creative Technology

IED Improvised Explosive Device

IWARS, Infantry Warrior Simulation

JSIMS Joint Simulation System

KC Kandahar City

MAS Multi-Agent Systems

MCCDC-OAD Marine Corps Combat Development Command, Operations Analysis Division

MC Monte Carlo

MDP Markov Decision Process

NMD Naturalistic Decision-Making

NOLH Nearly Orthogonal Latin Hypercube

NRC National Research Council

OCOMAS Organization Centered Multi-Agent Systems

OneSAF One Semi-Automated Forces

OODA Observe, Orient, Decide, and Act

PBC Perceived Behavioral Control

POMDP Partially Observable Markov Decision Process

PTSP Physical Traveling Salesman Problem

RL Reinforcement Learning

SOAR State, Operator, and Results

TD Temporal Differencing

TRAC TRADOC Analysis Center

TRAC-WSMR TRADOC Analysis Center -White Sands Missile Range

TPB Theory of Planned Behavior

TSP Traveling Salesman Problem

TTP Tactics, Techniques, and Procedures

RPD Recognition Primed Decision Making

SARSA State-Action-Reward-State-Action

SD Standard Deviation

SN Subjective Norm

USA United States Army

UCB Upper Confidence Bound

UCT Upper Control Bound for Trees

USMC United States Marine Corps

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGEMENTS

I have been blessed with a supportive family, committee, and advisor that have encouraged and supported me throughout the marathon of the last three years. My wife, Maurisa, patiently allowed me to continue "piddling on the computer" late into the evening without complaint. Katie, Jacob, Josh and Gabe never complained about their father's constant busy schedule. Katie excelled at school and sports throughout the process, making it hard for me to keep up with her on her 2-mile run by pushing her personal best down a little further each year. Without her leadership and support as the big sister in the house, my dissertation could not have been completed. Jacob continued to shine on the soccer field and in lacrosse, to set the standard in academic performance at school, and to serve as an excellent role model for his little brother. Joshua grew from a toddler to a young man - literally gaining at least a foot and a half of vertical in the process. His big heart and willingness to play at any time, and to forgive his dad for not having much time, inspired me to keep my nose to the grindstone. Gabriel joined the family after my first year in the program and watching him grow and learn has been a joy that I do not take for granted. I thank God everyday for placing me in the company of such a wonderful and supportive family and am humbled by the opportunity guide my children on the path to adulthood. They are my best friends. Maurisa is the love of my life and has been supportive every step of the way these last 18 years. I love her with all my heart.

Chris Darken has served as my mentor these last three years and as a role model for how to conduct research. His demeanor has always been positive and he constantly kept me moving forward. I cannot thank him enough for his patience and support. He is one of the most intelligent individuals I have ever had the pleasure of working with, but also one of the most personable. He is absolutely willing to meet each individual wherever they are and has the ability to read his audience to understand if they are still with him or if he needs to recast his material. This is a rare combination that serves him well as both a professor, advisor, and friend.

Michael McCauley was a constant source of sage counsel. His broad experience in academia and the applied world give him a unique perspective on research. His knowledge of experimental psychology is unmatched and his insistence on grounding research in empirical data when possible kept me from going astray on more than one occasion.

Arnold Buss is one of the most eminently practical and pragmatic gentlemen that I know. His insistence on fully defining models and objectives prior to touching code saved me countless hours, and had the potential to save me more had I been as disciplined as he. He is always willing to assist others regardless of the triviality of the question. His practical approach to modeling and simulation serve as an example to us all.

Jeffrey Appleget and Michael Jaye both served as anchors of support in all my efforts. Constantly positive in attitude and demeanor they expressed confidence throughout the process that I would finish, despite my own moments of doubt. I am indebted to both for their unwavering support and encouragement.

Finally, the strong team of analysts at TRAC-MTRY enabled me to wear my director hat part time and my student hat a majority of the time. Without this strong group stepping up and performing I would not have been able to complete my research on time. Jack Jackson has always been a source of support and encouragement and with him as the deputy I knew I could put things on autopilot as required. Jason Caldwell, Paul Evangelista, Tom Deavons, James Henry, Kevin Bolke, Ricky Brown, Ed Massotti, Steve Bitner, and Joe Vargas all rowed together as a team to make sure the organization continued to produce quality research. John Ruck and Harold Yamauichi consistently made themselves available to answer questions related to the implementation of my research into various models and went above and beyond to assist, as did Jimmy Liberato who configured and troubleshot our small Condor cluster, that enabled much of my benchmark testing. Imre Balough assisted in helping to demystify COMBATTXXI. Thanks to the students who allowed me to serve as their second readers and kept me motivated with their enthusiasm and ideas: Sotiris Papadoupolis, Dan McKaughn, Shawn Pollock, Ozcan Ozkan, and Brian Vogt.

This has been a growing experience that I will never forget and I thank my Lord and savior who was with me every step of the way. My faith assures me that I am walking the path he has laid out for me and I give all the glory to God for allowing me the opportunity to shine on his behalf.

> Even the youths shall faint and be weary, and the young men shall utterly fall:
>
> But they that wait upon the Lord shall renew their strength; they shall mount up with wings as eagles; they shall run, and not be weary; and they shall walk, and not faint. Isaiah, 40:30-31

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Modeling and simulation of military operations requires human behavior models capable of learning from experience in complex environments where feedback on action quality is noisy and delayed. This dissertation examines the potential of reinforcement learning, a class of learning algorithms from artificial intelligence, to address this need. Reinforcement learning provides an empirically developed conceptual model of human behavior to guide the implementation of a model for human decision making within simulation models (Lattal, 2010; Thorndike, 1911). Cognitive architectures are simulation-oriented models of individual human information processing and behavior often developed to emphasize specific aspects of cognition, depending on the use case (Langley, Laird, & Rogers, 2009; Zacharias, MacMillan, & Van Hemel, 2008). Autonomous agents are software agents designed to sense their environment and select appropriate actions to express in the environment-based on their perception of the state of the environment (Russell & Norvig, 2010). A simulation model is a representation of those elements of a real-world phenomena required to support either the issues for analysis or the training objectives of the user, dependent on the specified use case (Goerger, McGinnis, & Darken, 2005).

A. MOTIVATING PROBLEM

The Department of Defense of the United States relies on simulation models to inform decision-making processes for contingency planning and acquisition (Goerger et al., 2005; DMSO, 2004; Zacharias et al., 2008). Simulation models support analyses that provide insights into the combat effectiveness of weapon systems, the appropriate mix of capabilities, and the impact of tactics, techniques, and procedures (TTP) that would not otherwise be obtainable since, in many cases, physical experiments are infeasible or too costly. Training simulations serve to provide practice environments in which trainees can develop skill proficiency at relatively low cost and risk. The human decision maker is

represented in combat simulations with varying degrees of sophistication depending on the model (Zacharias et al., 2008). In many cases, agent decision-making is limited to a small set of reflexive rules, which limits the flexibility of these models and increases scenario development time and cost required to support analysis.

Many recent DoD studies center around capabilities designed to improve the quality and quantity of information provided to a decision maker (Zacharias et al., 2008). The analysis of the impact of information on combat operations requires an agent that, when presented with new or different sets of information from the environment, can change its behavior, choosing a different action-based on feedback from the environment. DoD training simulations require adaptive agents capable of providing trainees with realistic training experiences by serving as autonomous agents within the training environment, as well as software agents that can adaptively adjust the training environment in order to improve the trainees experience. DoD combat simulations need agents capable of learning from their experience in the environment and choosing different actions in response to different situations (Zacharias et al., 2008). The DoD analytic community also requires simulation agents whose decision-making is based on an empirically validated conceptual model and whose resultant behavior is explainable and traceable (J. Alt, Lieberman, & Blais, 2010; DMSO, 2004; Goerger et al., 2005; Zacharias et al., 2008). RL algorithms coupled with a model of human information processing, such as a practical cognitive architecture, provide one potential approach to address this need (J. K. Alt, Baez, & Darken, 2011; Anderson & Schunn, 2005; Zacharias et al., 2008; N. Taatgen, Lebiere, & Anderson, 2006).

The *temporal credit assignment problem* in RL refers to the challenge of assigning credit or blame to actions that lead to the achievement of a goal in sequential task settings. Complex tasks often require numerous steps to achieve a goal, and as an individual learns the correct sequence of steps the challenge becomes understanding which decisions should be reinforced to enable a shorter path in the next iteration. Various strategies exist for addressing this challenge, but it is still an open area of research within the RL community. The RL algorithm presented in this dissertation results in improved learning performance

over similar algorithms through its relatively straightforward, but novel, approach to the temporal credit assignment problem. This dissertation contributes insights in the application of RL to four use cases within military modeling and simulation.

B. DEFINITIONS AND TERMINOLOGY

This section provides a brief description of important definitions and terminology used in this dissertation. A *software agent* consists of a set of sensors, actuators, and an internal decision model (Russell & Norvig, 2010). The agent receives information regarding its environment in the form of *percepts* detected by its sensors. These percepts provide atomic-level information used by the agent to form a notion of *state*, which the agent uses to select an action to be expressed in the environment through its actuators, see Figure 1 (Russell & Norvig, 2010).

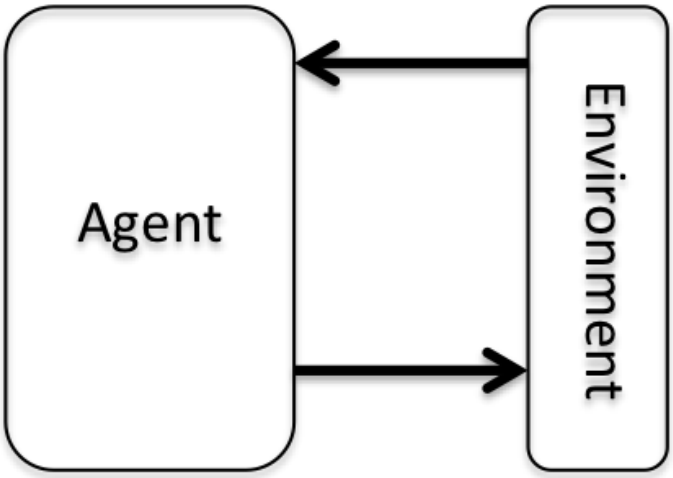
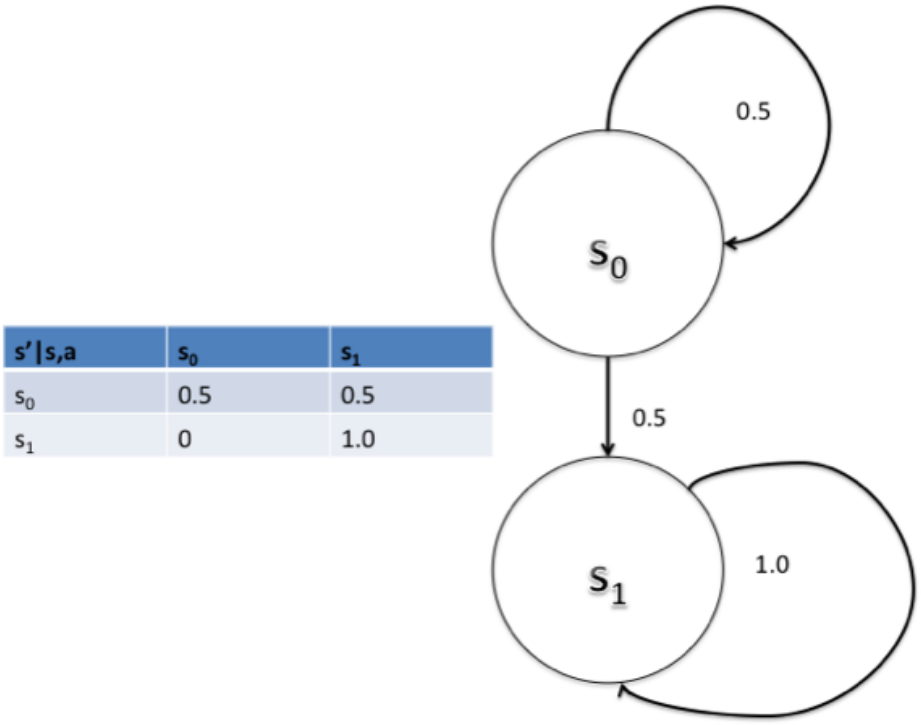


Figure 1: Agent environment interaction.

Assume that a software agent is placed in a *sequential decision problem* in a discrete state space, $s \in S$, and must choose from a finite set of actions, $a \in A$, so as to maximize its long-term reward, $V(s)$, defined as the sum of rewards provided from its reward function

which maps states to point rewards, $R(s) \rightarrow r$. If we assume the agent has access to a transition matrix, $P_a^{s,s'}$, describing the probability of transitioning from s given a to each of the possible states, s', then the environment is a fully observable environment (Russell & Norvig, 2010; Sutton & Barto, 1998). Further, this sequential decision problem is called a *Markov decision process* (MDP) since it adheres to the Markov property, see Figure 2.



s'|s,a	s_0	s_1
s_0	0.5	0.5
s_1	0	1.0

Figure 2: Simple two state MDP.

An MDP can be fully described by the tuple, $\langle S, A, P_a^{s,s'}, R \rangle$. A solution to an MDP, referred to as a policy, π, maps states to actions, $\pi(s) \rightarrow a$. If we assume that the preferences of the agent are stationary over-time then the accrual of long-term reward can be determined by a value-function, $V(s)$, which estimates the long-term value of a state using either an additive or discounted sum, where $\gamma \in [0, 1)$ is known as the discount factor and t is the trial count (Powell, 2011; Russell & Norvig, 2010; Sutton & Barto, 1998),

$$\sum_{t=1}^{h} r_t,$$

$$\sum_{t=1}^{\infty} \gamma^t r_t. \tag{1}$$

As $\gamma \to 0$, the agent places more weight on recent events and when $\gamma \to 1$ the discounted utility is equivalent to the additive case (Kaelbling, Littman, & Moore, 1996; Powell, 2011; Russell & Norvig, 2010). Each sequence of states is considered a unique policy, π. A policy is said to be *stationary* if it is deterministic and the action it chooses depends only on the current state, s (Ross, 1982). Since this satisfies the Markov property the sequence of states forms a Markov chain. Since the state transitions are Markovian it is appropriate to think of the value of a state, $V(s)$, in terms of expectations and the policy as a mapping to the probability of taking action a in state s. We will consider only the discounted case going forward and can state that the expected value of executing a policy, π, starting in state s is,

$$V^{\pi}(s) = E_{\pi}[\sum_{t=0}^{\infty} \gamma^t r_t | s_t = s]. \tag{2}$$

The optimal state-value function is the value that is the long-term reward the agent will gain if it starts in that state and executes the optimal policy, π^* (Kaelbling et al., 1996),

$$V^*(s) = \max_{\pi} E\{\sum_{t=1}^{\infty} \gamma^t r_t\}. \tag{3}$$

The optimal value function is uniquely defined and can be obtained through the solution of a set of simultaneous equations, referred to as the Bellman equations (Kaelbling et al., 1996),

$$V^*(s) = \max_{a}\{R_s^a + \gamma \sum_{s'} P_a^{s,s'} V^*(s')\}, \forall s \in S. \tag{4}$$

We can now determine an optimal policy, π^*, that yields the greatest expected long-term reward for a sequence of decisions (Kaelbling et al., 1996),

$$\pi^*(s) = \max_a \{R(s) + \gamma \sum_{s'} P_a^{s,s'} V^*(s')\}. \tag{5}$$

Assume that the agent has a fixed-time, $t \in T$, over which to maximize its long-term reward, a *finite horizon*, then π^* is said to be *nonstationary*, since it is a function of T (Powell, 2011; Russell & Norvig, 2010). If this time constraint is relaxed and the agent has an infinite amount of time over which to act, an *infinite horizon*, the optimal action depends only on the current state, s, and the optimal policy, π^*, becomes stationary.

The *value-iteration* algorithm, a technique from the field of dynamic programming, provides a means of solving the Bellman equations, stating that the utility of a state is equal to the immediate reward from the current state, s and the expected discounted utility of the next state, s', assuming that the agent chooses the optimal action available (Kaelbling et al., 1996; Ross, 1982; Russell & Norvig, 2010; Sutton & Barto, 1998),

$$V^\pi(s) \leftarrow R(s) + \gamma \max_a \sum_{s'} P_a^{s,s'} V^{n-1}(s). \tag{6}$$

This iterative form of the algorithm will typically proceed until it reaches an equilibrium-based on the maximum error allowed in the approximation of the value of a state, $V(s)$. Value-iteration serves as a basis for many of the RL algorithms. It also serves as the motivation for policy iteration, a technique to iteratively improve the performance of a policy by solving the analytic version of the Bellman equation and the action-values, $Q(s, a)$, associated with a given state, s (Dimitrakakis & Lagoudakis, 2008; Sutton & Barto, 1998),

$$Q^\pi(s, a) = R(s) + \gamma \sum_{s'} P_a^{s,s'} V^\pi(s'). \tag{7}$$

The action-value, $Q(s, a)$, on which we will focus much our our research, can then become the basis for determining the new policy, π,

$$\pi_{i+1}(s) = \max_{a} Q^{\pi_i}(s, a). \tag{8}$$

Knowledge of the transition probabilities, $P_a^{s,s'}$, is seldom available in real-world applications or to a human decision maker attempting to solve a complex MDP. The ability to learn the transition probabilities, $P_a^{s,s'}$, over-time, as is the case in model-based RL, has its own limitations due to the potentially large number of states and actions and the need to maximize utility while learning. Model-free methods of RL provide a useful alternative to determine the optimal policy, π^*, within sequential decision problems.

If we assume that the agent has a noisy sensor that affects its perception of information from the environment then the environment is no longer fully observable. The MDP now becomes a partially observable Markov decision process (POMDP) (Russell & Norvig, 2010). The elements of the MDP are still present, but now a sensor model is introduced. This noisy sensor and its impact on agent perception distinguish between MDPs and POMDPs. As a result, the agent looks for the optimal policy-based on its current perceived state rather than its actual state. RL techniques provide a feasible means of determining the optimal policy within a POMDP, but we will continue discussion of the RL problem in the context of an MDP often used in benchmarking the performance of RL systems, see Figure 3 (Sutton & Barto, 1998).

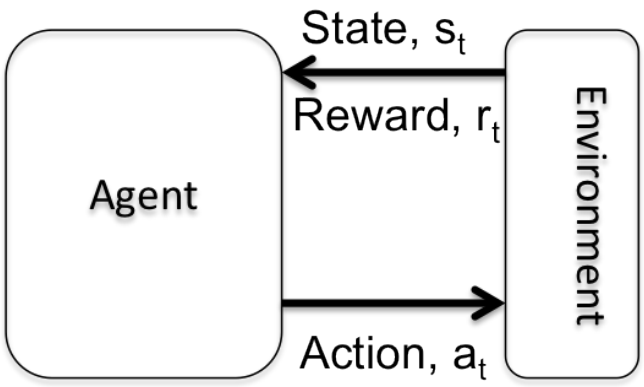


Figure 3: Agent-environment interaction in reinforcement learning.

A RL problem is a form of *goal-directed* learning from interaction, or trial and error, where the learner, an agent, tries to learn what actions to take to maximize the sum of a numeric reward signal, which might be immediate or delayed (Powell, 2011; Russell & Norvig, 2010; Sutton & Barto, 1998). RL problem formulations include three characteristics: sensation, action, and goal. An agent requires information regarding the state of the environment, the relevant action to affect the state, and a reward function mapping states to goals (Sutton & Barto, 1998).

A RL problem in which the next state, s', only depends on the current state, s, that satisfies the Markov property, is an MDP, but RL is not limited to MDPs, though MDPs often serve as benchmarking environments for RL algorithms. RL can be successfully applied to non-Markov environments. RL tasks can be either episodic, with finite horizon, if the task breaks into natural subsequences, referred to as *episodes*, or continuing tasks, infinite horizon, if the task goes on continually without limit, with the long-term cumulative reward being treated in either an additive or discounted manner as described previously. Episodes end in unique *terminal* states, such as when a goal is achieved.

An important distinction between techniques used to solve MDPs and the RL problem is the degree of information available to the agent. *The RL agent requires no apriori*

model of the environment or the reward function, $R(s)$. Instead it must learn through trial and error as Russell and Norvig summarize nicely,

> Imagine playing a new game whose rules you don't know; after a hundred or so moves, your opponent announces, "You lose." This is RL in a nutshell. (Russell & Norvig, 2010)

The core components of a RL system, as defined by Sutton and Barto, are a policy, $\pi(s)$, a reward function, $R(s)$, and a value function, $V(s)$ or the value of a an action taken in a given state, $Q(s, a)$ (Sutton & Barto, 1998). Some systems make use of an internal learned model of the environment, but here we scope our research to model-free methods.

Model-free methods seek to estimate the value of a state, $V(s)$, or the value of a state-action pair, $Q(s, a)$, and have proven successful in a range of application areas across several disciplines (D. Bertsekas & Tsitsiklis, 1996; D. Bertsekas, 2011; Russell & Norvig, 2010; Powell, 2011).

Model-based approaches refer to the class of RL approaches that maintain an internal representation of the transition matrix describing its environment. These approaches attempt to learn the value of a state, $V(s)$, and the transition probabilities, $P_a^{s,s'}$, which often results in lengthened learning-times. The policy, $\pi(s)$, maps states to actions, $\pi(s) \rightarrow a$, as discussed previously. Reward functions provide the agent with a numeric signal when the goal has been obtained by mapping perceived states to a reward, $R(s) \rightarrow r$. The value function, $V(s)$, or action-value function, $Q(s, a)$, tracks the long-term value of a state or state-action pair, (s, a), using either additive or discounted methods.

RL algorithms are often categorized by the method that they employ to form the estimate of the value of a state-action pair, (s, a), with the two main categories being *Monte Carlo (MC)* and *temporal differencing (TD)* methods. While both methods require samples of each state-action pair, (s, a), and the rewards, r, gained from experience with the environment, TD methods refer to those methods that form an expectation rather than learning directly (Szepesvari, 2010).

Two classic problems present themselves in RL, the *exploration-exploitation dilemma* and the *temporal credit assignment problem*. The exploration-exploitation dilemma refers to the problem of determining when to choose an action perceived as non-optimal. If an agent always chooses a greedy option, it has a stationary greedy policy, then it risks missing out on a potentially higher reward from an untried (s, a). It must balance this against the need to maximize long run expected utility. This problem occurs not only in RL, but in sequential decision making problems in general (Robbins, 1952; Ross, 1982). We will discuss several non-stationary, stochastic, policy strategies to address this problem.

The temporal credit assignment problem refers to the challenge of efficiently assigning rewards to state-action pairs when rewards are delayed (Szepesvari, 2010). This challenges appears in tasks requiring the completion of multiple steps prior to reach the goal such as in a path finding task. How does one identify which actions attempted during the episode contributed to the achievement of the goal?

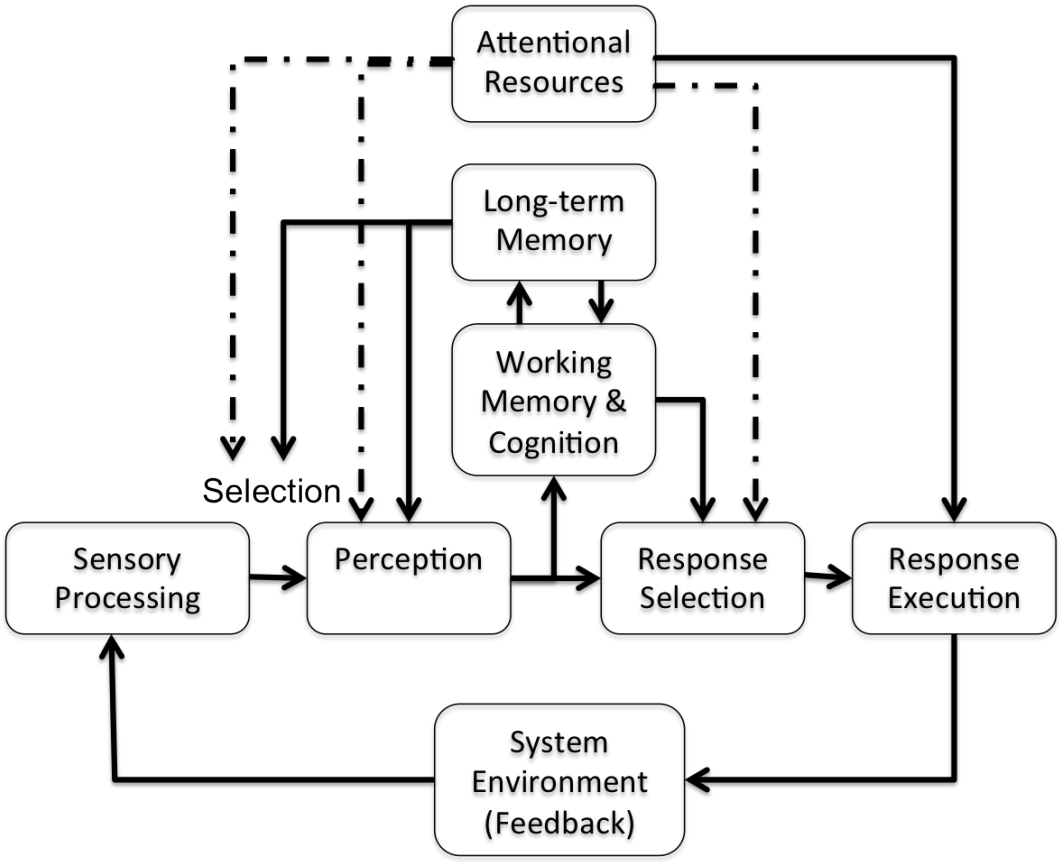


Figure 4: Information processing model of human cognition.

Cognitive architectures are simulation-oriented models of individual human information processing and behavior that emphasize specific aspects of cognition, depending on the use case (Langley et al., 2009; Zacharias et al., 2008). The National Research Council identifies cognitive architectures for their relevancy to three core areas of military modeling: analysis and forecasting in planning, simulation for training, and design and evaluation for acquisition (Zacharias et al., 2008). Cognitive architectures typically incorporate some derivative of the human information processing model, see Figure 4, which is minimally described as containing functions for perception, sense-making, meta-cognition, long-term memory, and action-selection (J. K. Alt et al., 2011; Klein, 1993; Wickens & Hollands, 2000). The incorporation of RL within existing cognitive architectures, such as ACT-R

and SOAR, has been positively received and expanded the flexibility of these architectures, showing promise in developing adaptive agents that provide credible representations of human behavior within complex DoD simulation models (Laird, 2008; Laird & Wray III, 2010). Within the DoD modeling, simulation, and analysis communities, models of decision making such as a recognition prime decision making (RPD) and Boyd's observe, orient, decide, and act (OODA) loop have gained credibility as representing the essential functions of military decision making (Zacharias et al., 2008; Klein, 1993).

One challenge in the use of existing cognitive architectures in military modeling and simulation is the data required to populate these models, whose intended use lies in understanding and representing detailed cognitive processing of human cognition at a fine-grained-level on specific tasks. The assumptions that must be accepted and the challenges of collecting data to populate these models make them challenging to incorporate into DoD analytic models due to the strict model and data validation requirements of these models (Cassenti, 2009). A practical cognitive architecture that streamlines data required while still meeting the functional requirements to support traceable autonomous agents within military modeling and simulation is required (J. K. Alt et al., 2011).

C. CHALLENGES OF ADAPTIVE AGENT DECISION MAKING IN ANALYTIC AND TRAINING SIMULATIONS

This section illuminates the challenges of using adaptive agents in simulation models intended for analytic and training purposes. Simulation model development and subsequent analysis of the simulation output cannot be treated as independent activities. The simulation model must support providing insights into the issues for analysis identified by the analyst. The analyst must develop the simulation model and situational vignettes that represent the measurement space required to inform the analysis question. Many combat models used by the analytic community, such as COMBATXXI, IWARS, and OneSAF, provide the user the ability to construct combat scenarios and assign behaviors to agents within the simulation (DA, 1999; DMSO, 2004; Goerger et al., 2005). These models rep-

resent the physics of combat well, but a common shortfall exists in the representation of human decision-makers. Each model provides a rule-based mechanism for controlling behavior, but suffers from a brittleness problem when the agent encounters situations that were not anticipated apriori. Agents designed for use in training simulations encounter the same challenges, but brittleness problems become more readily apparent due to the interaction with the trainee and can serve to distract the trainee from the intended training objectives. Other DoD models, such as the Cultural Geography (CG) model, are designed to represent aspects of the civilian population rather than combat forces. These agents represent not a single human, but a population segment. Similar limitations in the behavior of agents exist in this class of simulation model (J. Alt, Jackson, Hudak, & Lieberman, 2009).

Adaptable agents, such as those based on RL, provide a reasonable option to avoid the brittleness problem and to potentially reduce the overhead associated with scenario development and are based on an empirically derived conceptual model that is readily explainable (Zacharias et al., 2008). Many issues arise in the use of adaptive agents within DoD simulation models intended for analysis such as: (i) is the agent decision model-based on a valid conceptual model of human decision-making?, (ii) what are the data requirements to populate the model and is it feasible to obtain from a valid source?, and (iii) are the results reproducible? These issues often serve as an impediment to the use of adaptive agents within analytic models, though there is general consensus that such agents would improve the efficiency of scenario development processes and the value of simulation analysis, particularly those analysis products intended to gage the value of capabilities designed to enable greater access to information. This dissertation stresses the need to utilize *explainable and practical* approaches to the use of adaptive agents with these issues in mind.

D. DISSERTATION CONTRIBUTIONS AND ORGANIZATION

The primary contribution of this research is the novel use of the exponentially weighted average reward as an action-value estimator in reinforcement learning systems

in order to address the temporal credit assignment problem in reinforcement learning. This relatively straightforward approach improves learning speed over dominant existing approaches in task environments with noisy and delayed reward signals and improves performance in non-stationary environments, while reducing the number of parameters required to be specified by the system designer from 3, for current dominant approaches, to 2. Many real-world applications fall into this category of problem and in these cases delays in learning or recovery can result in control system failures and lost resources. The results provide the modeling and simulation community with a method that speeds learning in these challenging cases, reducing the time required to train autonomous software agents and the time required for agents to adjust to changes in their environment. These performance results carry over into each of the multiple modeling and simulation application areas examined in this research.

A second major contribution of this research is a novel methodology and example application of the use of reinforcement learning as a means of verifying the reward structure of a training simulation. The reward structure in a training simulation directly impacts trainee learning-time and outcomes. A weak reward signal will result in slower learning and a reward signal that rewards trainee actions that are not consistent with learning objectives will result in the trainee learning the wrong objectives. This research demonstrates the use of reinforcement learning to examine the reward structure and produce an example of the learned behavior, or policy, that can provide the training simulation designer feedback on the student behaviors rewarded by the training simulation prior to the simulation ever touching student hands, allowing the developer the opportunity to identify and correct deficiencies prior to fielding.

A third major contribution of this research is a methodology and application of reinforcement learning to address limitations of a discrete event simulation. This simulation is used to produce a feasible schedule for unmanned aerial assets that maximizes a value function by correctly pairing platforms with mission demands in the context of a combat scenario. The current approach employs a linear program that maximizes value over a

finite-time horizon, but fails to provide a feasible schedule that maximizes value in cases where high value targets appear beyond the time horizon or where high value emergent targets become available following the initial allocation. Further, this approach fails to learn to recognize the cues leading to these situations, as a human decision maker would over-time, and makes these mistakes consistently resulting in feasible, but non-value maximizing schedules. The value of the feasible schedule produced in different combat simulations for a given mix of unmanned platforms is used to inform acquisition decisions regarding unmanned assets, so the current tools limitations directly impact the representation of the value of a given mix to senior decision makers. Since the difference in the value lost to these cases varies across mixes, the analyst cannot know how this systematic issue effects results in the aggregate. This research demonstrates the use of reinforcement learning to address these cases and an approach that relaxes the strict requirement for fully observable demands currently imposed on the simulation.

A fourth major contribution of this research a methodology and application of reinforcement learning to represent human decision-making within a combat simulation. This straight-forward approach provides an empirically developed conceptual model of human decision making, important for eventual model validation, that facilitates dynamic decision making and allows agents to learn from interaction with their environment. This approach incorporates the novel use of reinforcement learning within hierarchical task networks, providing the potential to enable adaptive decision making within complex behaviors. This has particular relevance for enabling agents that adapt to the behavior of an opposing force, as human decision makers do, as opposed to relatively brittle scripted methods currently in use.

A fifth contribution of this research is the development and application of a novel practical cognitive architecture that facilitates the representation of human information processing and the inclusion of domain knowledge in a structured manner that enables the selective use of goal-driven reinforcement learning to represent human decision making. The cognitive architecture provides an understandable framework to incorporate the effects of

perception, working memory, and dynamic goal-setting within simulation agents. This is particularly relevant for analysis topics related to the value of information or the impact of networked sensors. The cognitive architecture also has relevance to the representation of civilian behavior in conflict areas, where the analysis focuses on the beliefs and interests of a population and the cognitive architecture provides a organizing construct. This contribution was incorporated into a social simulation used to facilitate war-games that received a 2011 Army Modeling and Simulation Office award for excellence in analysis (http://www.ms.army.mil/about/awards/fy11-awards.html).

The flow of the dissertation is as follows: Chapter II provides a review of reinforcement learning and cognitive architectures. Chapter III specifies and documents the performance of a the novel use of an exponentially weighted average that takes advantage of continuous-time as an action-value estimator in a reinforcement learning system in order to addresses the temporal credit assignment problem in an intuitive manner. Chapter IV details the application of RL to each of the applied military simulation use cases. Chapter V details the development and use of the cognitive architecture within a military social simulation. Chapter VI provides a summary of contributions and describes the path forward for future research.

II. REINFORCEMENT LEARNING AND COGNITIVE ARCHITECTURES FOR AUTONOMOUS AGENT DECISION-MAKING

> When an organism acts upon the environment in which it lives, it changes that environment in ways which often affect the organism itself. Some of these changes are what the layman calls rewards, or what are now generally referred to technically as reinforcers: when they follow behavior in this way, they increase the likelihood that the organism will behave in the same way again. (Ferster & Skinner, 1957)

This chapter contains the foundations for the subsequent development of a new action-value estimator, describes measures by which the performance of RL algorithms are typically judged, and provides foundational material for the subsequent development of a practical situation-based cognitive architecture.

The simulations that DoD analysts use to inform decisions are typically large and complex, with a large potential state space (Cioppa, Lucas, & Sanchez, 2004). The agents within these models typically make decisions based on simple decision rules or custom scripting for each scenario making code or behavior reuse difficult. These hard decision rules lock agents into set TTPs for a given decision situation. This makes the exploration of new capabilities, which potentially change TTPs, difficult and time-consuming. These relatively brittle decision mechanisms also preclude the representation of differing levels of experience and training present in real human decision makers (Zacharias et al., 2008). Adaptive agents are required to provide simulation models capable of representing the nuances of human decision makers in a more credible manner. Adaptive agents are needed that possess the following characteristics:

- Empirically derived conceptual model of decision-making.
- Capable of adapting behavior gracefully when environment changes.
- Learn appropriate behavior in minimum time and with minimum error.

- Require few tunable parameters and initialization data.
- Behavioral changes occur in a transparent and explainable manner.

A variety of approaches for employing adaptive software agents exist in the literature, but few have been explored within DoD simulation models (Zacharias et al., 2008; Shoham & Leyton-Brown, 2009). Several model-free RL methods have been proposed and documented in the RL literature including Q-learning and SARSA, but challenges in learning with noisy or delayed rewards still exist (Powell, 2011; Sutton & Barto, 1998; Szepesvari, 2010). Cognitive architectures of varying degrees of complexity have been applied to this problem as well, such as ACT-R, SOAR, CLARION and PMFServ, though not to the the specific challenges of representing human decision makers in DoD analytic simulation models (Langley et al., 2009; Sun, 2007b, 2006, 2007a; N. Taatgen et al., 2006; Laird, 2008; Wray & Jones, 2006). The most promising of the current approaches, in terms of satisfying the characteristics described above, are model-free RL methods and information processing based models of cognition that leverage domain knowledge, such as cognitive architectures.

A. REINFORCEMENT LEARNING

> Reinforcement learning is learning what to do how to map situations to action so as to maximize a numerical reward signal. (Sutton & Barto, 1998)

This section provides a review of RL literature from the field of artificial intelligence to include a brief summary of literature justifying its use as a conceptual model for human behavior representation. The relevance of RL as a foundation of human behavior and the broad utility of the algorithmic approaches developed in the artificial intelligence community in solving a applied problems from a variety of fields, including dynamic programming and operations research, are emphasized (D. Bertsekas & Tsitsiklis, 1996; Powell, 2011).

RL within the artificial intelligence community was originally inspired by research in animal psychology and Thorndike's "Law of Effect," describing the effect of rewards

on an animal's likelihood of selecting actions, still widely regarded as a basic foundational principle accounting for much of animal, including human, behavior (Lattal, 2010; Thorndike, 1911).

> Of several responses made to the same situation, those which are accompanied or closely followed by satisfaction to the animal will, other things being equal, be more firmly connected with the situation, so that, when it recurs, they will be more likely to recur; those which are accompanied or closely followed by discomfort to the animal will, other things being equal, have their connections with that situation weakened, so that, when it recurs, they will be less likely to occur. The greater the satisfaction or discomfort, the greater the strengthening or weakening of the bond. (Thorndike, 1911)

Thorndike's research on the Law of Effect influenced Skinner's research in operant conditioning (Skinner, 1938). This research is widely documented and empirically derived, making it a feasible candidate for a conceptual model for human behavior representation in DoD simulation models. Acquisition of responses to stimulus by humans is still an active area of research in the experimental psychological and neuroscience communities (Lattal, 2010; Nargeot & Simmers, 2011; Okouchi, 2009). The behavioral economics community, particularly those who study decision-making under uncertainty, has also continued empirical work in this area (Sundaram, 2005; Yi, Steyvers, & Lee, 2009). Duffy provides a summary of recent empirical work in the area of agent-based computational economics (Duffy, 2006). Several recent experiments have compared the results of human decision-making experiments with the results produced by popular RL methods providing support to the notion that RL can provide insight into human behavior in certain decision situations (Acuña & Schrater, 2010; Ishida, Sasaki, Sakaguchi, & Shimai, 2009; Nedic, Tomlin, Holmes, Prentice, & Cohen, 2008; Steyvers, Lee, & Wagenmakers, 2009; Walsh & Anderson, 2010; Lee, Seo, & Jung, 2012).

Reinforcement learning contributes to the theory of planned behavior which has been empirically studied as an explanation for behavior adoption (Ajzen, 1991). The theory of planned behavior places reinforcement learning in a social context, by incorporating societal norms, perceived behavioral control and attitude into the reward signal associated with a potential behavior.

Reinforcement learning also overlaps with naturalistic decision-making (NDM), particularly Klein's recognition-primed-decision-making (RPD) model (Klein, 1993, 2008). This model, based on empirical observations of decision-makers in primarily military and emergency services organizations, describes pattern-matching that goes on with expert decision-makers. Experts tend to recognize situations more readily than others, allowing them to identify the appropriate action to take based on their experiences that led to successful outcomes when previously in similar situations.

Most recently, studies utilizing brain imaging technologies have identified patterns of reinforcement at the neuronal-level within the brain (Lee et al., 2012). Taken as a whole, this body of work provides an empirically developed conceptual model of individual human behavior, a requirement for DoD simulation models, upon which to base our implementation (DMSO, 2004).

Sutton and Barto provide a full chronological account of the impact of RL on the field of artificial intelligence (Sutton & Barto, 1998). Since the sequential decision problem arises in multiple applied settings, multiple fields developed techniques to address these issues with several disciplines arriving at similar solutions. Operations research developed the Bellman equations, which led to the development of the field currently referred to as dynamic programming, to address the optimal control problem (Powell, 2011). During the 1980s, the RL community and dynamic programming community connected with Watkin's Q-learning algorithm (Powell, 2011; Sutton & Barto, 1998; Watkins & Dayan, 1992). A third line of effort began under those who study control theory, motivated by physical operating processes in continuous-time, with continuous states and actions. This led to approximate dynamic programming, initially called heuristic dynamic programming. The

connections between this branch of research and RL were highlighted and enumerated by Bertsekas, who reclassified RL as neuro-dynamic programming (D. Bertsekas & Tsitsiklis, 1996; D. P. Bertsekas, 1995; Powell, 2011). Powell attempts to unite neuro-dynamic programming, RL, and dynamic programming as approximate dynamic programming (Powell, 2011). This research is primarily focused on the RL problem as defined by Kaelbling et al. and Sutton and Barto and the algorithms and techniques developed and applied in the *artificial intelligence* community (Kaelbling et al., 1996; Sutton & Barto, 1998).

The exploration-exploitation dilemma refers to the need to balance the ratio of exploratory actions and greedy action in order to maximize long-term reward. Since the RL agent must explore its environment to understand its reward structure, but is also tasked with maximizing its long-term reward, balancing the ratio between greedy and exploratory actions is of critical importance.

The temporal credit assignment problem refers to the need to efficiently assign rewards to state-action pairs when rewards are delayed, as in path finding tasks. Delayed rewards make it difficult to determine which of the state-action pairs attempted during the episode contributed to the achievement of the goal (Sutton & Barto, 1998).

Dominant approaches to this problem rely on hill-climbing approaches known as temporal-differencing (TD) methods that can be slow to learn, especially in environments with noisy reward signals. In order to address this problem, a family of value functions that incorporate a bias parameter in conjunction with standard TD methods tend to dominate the literature. These techniques require the setting of three tunable parameters, making their implementation challenging, and still often result in slow learning-times. These techniques do not explicitly take into account environment time, the time scale that controls the dynamics of the environment in which the RL agent is operating, making them less sensitive to changes in the environment that can affect performance.

In the remainder of this section, we provide additional background on temporal-differencing methods (TD), Monte-Carlo methods (MC), and temporal-differencing with a bias parameter (TD(λ)) methods, and detailed coverage of four relevant model-free methods.

1. Temporal Differencing, Monte-Carlo, and TD(λ) Methods

Approaches to model-free learning are broadly characterized as Monte Carlo, MC, or temporal-differencing, TD, methods, with TD(λ) bridging the two approaches by incorporating the notion of eligibility traces to address the credit assignment problem (Sutton & Barto, 1998; Russell & Norvig, 2010; Powell, 2011). Model-free methods estimate the value of a state, $V(s)$, or the value of a state-action pair, $Q(s, a)$,–the focus of our discussion from here forward–without the need for an internal model of the environment's dynamics and its associated learning cost.

MC methods typically have to wait until the end of an episode and then update the value of all state-action pairs, (s, a), visited, while TD methods make updates as observations occur using a noisy hill-climbing approach. It is still an open question as to which techniques, MC or TD converge faster, though both have been shown to converge to true estimates of the target value and the optimal policy under asymptotic conditions seldom present in real-world applications (Sutton & Barto, 1998). MC methods sample returns from completed *episodic* tasks to estimate the value of a state, $V(s)$, as described by Sutton (Sutton & Barto, 1998). First-visit MC and every-visit MC are two well-documented techniques, which recursively update the sample mean for each state using either first-visit or every-visit updating following the end of an episode, with $\alpha_t = \frac{1}{t}$, where i is the count of the visits to a state,

$$V(s) = V(s') + \alpha_i(r - V(s')). \tag{9}$$

First-visit MC updates this general equation using information from only the first visit to each state during an episode, while every-visit MC uses information available from

each visit during an episode with discounting incorporated most recently by Szepesvari, see Algorithm 11 and Algorithm 12 in Appendix A (Szepesvari, 2010).

MC methods' ability to learn directly by sampling from the environment make them attractive, but can result in longer learning-times than TD algorithms, however, conceptualization of sequential decisions problems as a series of bandit problems has led to their use in planning algorithms, such as Upper Control bound for Trees (UCT) (Kocsis & Szepesvári, 2006; Sutton & Barto, 1998). The relative simplicity of these techniques makes them attractive for a variety of applications, though they still make use of a hill-climbing process in the update (Bouzy & Chaslot, 2006; Valgaeren, Croonenborghs, & Colleman, 2009; Szita, Chaslot, & Spronck, 2010; Asmuth & Littman, 2011). TD RL methods combine MC methods and those from dynamic programming to estimate the value of a state (Sutton & Barto, 1998; Kaelbling et al., 1996). Methods that make use of eligibility traces, such as TD(λ), further combine TD and MC methods resulting in a class of methods with improved learning rate and less bias than pure TD methods according to the literature (Sutton & Barto, 1998; Szepesvari, 2010; Powell, 2011). Note that the methods discussed so far all typically are applied to learn the value of a state rather than a state-action pair. An eligibility trace, as defined by Sutton, tracks the *eligibility* of a state, or state-action pair, to receive credit for future rewards. These techniques require the use of a third parameter, λ in addition to the discount rate, γ, and the learning rate, α.

TD methods update the value of each state-action pair, $Q(s, a)$, using the most recent reward, r, and the estimated value of the next state-action pair, $Q(s', a')$, rather than waiting until the end of the episode (Sutton & Barto, 1998). TD(λ) combines elements of MC and TD methods into a single framework to estimate the value of each state, $V(s)$, through the use of eligibility traces, see Algorithm 13 in Appendix A (Sutton & Barto, 1998; Szepesvari, 2010).

In order to apply TD techniques to determine the value of a state-action pair we rely on algorithms such as Q-learning and SARSA, two dominant model-free algorithms from this class. In order to apply TD(λ) techniques to estimate the value of a state-action pair,

we rely on the extensions of these algorithms which incorporate the notion of eligibility traces, Q(λ) and SARSA(λ). We provide a brief review of these four TD methods in the next section.

a. Q-learning and Q(λ)

Q-learning is an off-policy TD approach requiring no internal model of the environment often used in RL applications. Let,
$\delta = r' + \gamma max_a Q(s', a) - Q(s, a)$,

$$Q(s, a) \leftarrow Q(s, a) + \alpha\delta, \tag{10}$$

where γ is a discount factor and α is referred to as a learning rate or step size parameter, see Algorithm 1.

Q-learning approximates the optimal action-value function, Q^*, regardless of the policy, π, as long as each state-action pair can be visited an infinite number of times (Watkins & Dayan, 1992). This relies on the Robbins-Monro conditions, regarding the size and sequence of the learning rate parameter, α, stated here: $\sum_{k=1}^{\infty} \alpha_k = \infty$ and $\sum_{k=1}^{\infty} \alpha_k^2 < \infty$. Convergence to the optimal policy, π^*, is guaranteed for the sample average case, where the learning rate $\alpha_k = \frac{1}{k}$, but not for constant learning rate, α. Q-learning is known to systematically overestimate the value of the optimal action in a given state, Q^*, but remains an important algorithm due to its simplicity and effectiveness, which in applications where precise estimate are not as important as simply identifying the optimal policy, π^*, is not impacted by this overestimation (Thrun & Schwartz, 1993).

The convergence rate of Q-learning is very dependent on the sequence of α. A typical modification to ensure convergence is to gradually decay the learning rate as a function of time or samples. Note that Sutton makes a claim that the lack of convergence is a benefit in non-stationary environments and that a fixed, but properly tuned, constant value of α is required for practical applied work (D. P. Bertsekas, 1995; Sutton & Barto, 1998).

Kaelbling notes that although the algorithm seems to be the most effective for delayed reward domains it does not address issues regarding large state spaces and convergence time can be long (Kaelbling et al., 1996; Szepesvari, 2010). Though one of the first model-free algorithms proposed, Q-learning's importance as a benchmark algorithm, and its utility in application, are demonstrated through its continued application and its many extensions (Powell, 2011; Sutton & Barto, 1998; Szepesvari, 2010). As late as 2011, Yu and Bertsekas proposed a new version of Q-learning intended for use in the stochastic shortest path problem with reduced computational cost due to the incorporation of a solution of the optimal stopping problem (Yu & Bertsekas, 2011).

Algorithm 1 Q-learning

1: Parameters: s, the current state; Q(s,a), current value estimate of a state-action pair; γ, discount rate; α, learning rate.
2: Initialize $Q(s, a)$ arbitrarily.
3: Initialize s.
4: Return a using $\pi(s)(ie.\epsilon - greedy, \beta)$.
5: **for** For t in T: **do**
6: Take action a, observe r', s'.
7: Return a' using $\pi(s')(ie.\epsilon - greedy, \beta)$.
8: $\delta = r' + \gamma max_a Q(s', a) - Q(s, a)$
9: $Q(s, a) \leftarrow Q(s, a) + \alpha\delta$
10: **end for**
11: Increment t. If $t \leqslant T$, go to line 3.

$Q(\lambda)$ combines Q-learning with eligibility traces, with alternative versions from Peng and Watkins. Focusing on Watkin's version, the general idea is to reset the eligibility trace, $e(s, a)$, for each exploratory action, which can result in poor performance when the ratio of exploratory actions is large (Sutton & Barto, 1998). The eligibility trace, e(s,a), for the greedy case is incremented by one for the chose action as described previously, but the value is only decayed by λ if the action chosen is the optimal greedy action.

Algorithm 2 Q(λ)

1: Parameters: s, the current state; Q(s,a), current value estimate of a state-action pair; e(s,a), current eligibility to receive credit of current state-action pair; γ, discount rate; α, learning rate; λ, decay rate.
2: Initialize $Q(s, a)$ arbitrarily.
3: Initialize s.
4: Return a using $\pi(s)(ie.\epsilon - greedy, \beta)$.
5: **for** For t in T: **do**
6: Take action a, observe r', s'.
7: Return a' using $\pi(s')(ie.\epsilon - greedy, \beta)$.
8: $\delta = r' + \gamma max_a Q(s', a) - Q(s, a)$
9: $e(s, a) \leftarrow e(s, a) + 1$
10: **for** all (s,a): **do**
11: $Q(s, a) \leftarrow Q(s, a) + \alpha\delta e(s, a)$
12: **if** $a = a^*$: **then**
$e(s, a) \leftarrow \gamma\lambda e(s, a)$
13: **else**
$e(s, a) \leftarrow 0$
14: **end if**
15: **end for**
16: **end for**
17: Increment t. If $t \leqslant T$, go to line 3.

Otherwise the eligibility trace is set to zero, e(s,a)← 0, prior to the next decision cycle, see Algorithm 2.

b. SARSA and SARSA(λ)

SARSA (State-Action-Reward-State-Action) is an on-policy TD method distinguished from Q-learning by the timing of the backup. Q-learning backs up the best estimate of the value of a state-action pair, $Q(s, a)$, during each observation period, while SARSA backs up the value of each state action pair, $Q(s, a)$, following the execution of each action (Powell, 2011). SARSA is generally considered less flexible than Q-learning (Russell & Norvig, 2010). SARSA shares the same limitations in regard to the choice of learning rate parameter, α, as described above for Q-learning. Here let

Algorithm 3 SARSA

1: Parameters: s, the current state; Q(s,a), current value estimate of a state-action pair; γ, discount rate; α, learning rate.
2: Initialize $Q(s, a)$ arbitrarily.
3: Initialize s.
4: Return a using $\pi(s)$ (ie. ϵ-greedy, β).
5: **for** t in T: **do**
6: Take action a_t, observe r_{t+1}, s_{t+1}.
7: Return a_{t+1} using $\pi(s_{t+1})(ie.\epsilon - greedy, \beta)$.
8: $\delta = r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)$
9: $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\delta$
10: **end for**
11: Increment t. If $t \leqslant T$, go to line 3.

$$\delta = r' + \gamma max_a Q(s', a') - Q(s, a),$$

$$Q(s, a) \leftarrow Q(s, a) + \alpha\delta, \tag{11}$$

and we see that the difference between Q-learning and SARSA is in the δ term only. SARSA assumes the agent is in state s, chooses action a according to the policy being followed, $\pi(s)$, observes a reward, r, and the next state, s', prior to choosing a second action a based on the same policy, see Algorithm 3. Bertsekas equates SARSA with a variant of optimistic policy iteration with no particular convergence guarantees and Powell describes SARSA as oscillating about the true estimate in most cases with no guarantees on error bounds (D. Bertsekas & Tsitsiklis, 1996; Powell, 2011).

SARSA uses the same policy to decide which action to evaluate and to choose the next action one step in the future. Powell refers to the policy that determines the action to take as a behavior policy in physical systems and a sampling policy in a simulation setting. The policy that determines the action that is best is the target policy in RL, he refers to it as the learning policy. SARSA combines learning and policy sampling resulting in on-policy learning, while Q-learning employs different learning and sampling policies (Powell, 2011).

SARSA(λ) uses eligibility traces, $e(s, a)$, as we saw in Q(λ), but it does not reset traces on exploratory actions. This leads to reports of better empirical performance than $Q(\lambda)$, see Algorithm 4. All the estimators we have discussed must be paired with a policy that balances the exploration and exploitation trade-off. There are no convergence guarantees associated with SARSA(λ) or Q(λ) (Sutton & Barto, 1998).

Algorithm 4 SARSA(λ)

1: Parameters: s, the current state; Q(s,a), current value estimate of a state-action pair; e(s,a), current eligibility to receive credit of current state-action pair; γ, discount rate; α, learning rate; λ, decay rate.
2: Initialize $Q(s, a)$ arbitrarily, $e(s, a) = 0, \forall(s, a)$.
3: Initialize s.
4: Return a using $\pi(s)(ie.\epsilon - greedy, \beta)$.
5: **for** t in T: **do**
6: Take action a, observe r', s'.
7: Return a' using $\pi(s')(ie.\epsilon - greedy, \beta)$.
8: $\delta = r' + \gamma Q(s', a') - Q(s, a)$
9: $e(s, a) \leftarrow e(s, a) + 1$
10: **for** all (s,a): **do**
11: $Q(s, a) \leftarrow Q(s, a) + \alpha\delta e(s, a)$
12: $e(s, a) \leftarrow \gamma\lambda e(s, a)$
13: **end for**
14: **end for**
15: Increment t. If $t \leqslant T$, go to line 3.

The four RL algorithms reviewed here, Q-learning, Q(λ), SARSA, and SARSA(λ), provide a representative sample of the dominant methods employed in the model-free estimation of the value of a state-action pair in practice. They all make use of a noisy hill-climbing approach, which can result in slowed learning in noisy or delayed reward environments. The use of a bias parameter, λ, to control bias through the use of

eligibility traces, adds another parameter that must be calibrated in addition to the discount rate and the learning rate. These methods also do not make use of continuous-time information from the environment within which the RL agent is operating, instead simply relying on an action counter. This raises the question of whether a more straightforward approach that takes advantage of the historical average reward and makes better use of environment time could result in faster learning, particularly in environments with noisy and delayed rewards.

2. Exporation and Exploitation

The exploration-exploitation trade-off presents itself in many applied sequential decision making applications (sequential sampling, control theory, dynamic programming) (Robbins, 1952; D. Bertsekas & Tsitsiklis, 1996; Sutton & Barto, 1998; Powell, 2011). The exploration-exploitation dilemma in RL refers to the challenge of knowing when to choose an action that does not adhere to the optimal, or greedy, policy. If an agent placed in an unknown MDP adheres to a stationary policy that maximizes its long-term reward it runs the risk of trying one action, receiving a single reward and then cycling on that action since it will be the only action associated with a reward (Powell, 2011; Sutton & Barto, 1998). An agent following such a strategy could miss out on potentially greater reward to be achieved by attempting previously unexplored actions. This need to explore the state-space is a requirement for RL systems to function effectively. Robbins described this problem as a sequential sampling problem and is familiar to those interested in control theory as well as dynamic programming (Robbins, 1952).

The policy selected to address this issue impacts the quality of the estimate provided by the value function since it controls sampling. Stochastic policies, such as ϵ-greedy and Boltzmann-exploration, referred to in this research later for brevity as β, guarantee the exploration of the full state-action space as $n \to \infty$, with variations that allow the ratio to respond dynamically to the environment (Sykulski, Adams, & Jennings, 2010; Tran-Thanh, Chapman, Munoz De Cote Flores Luna, Rogers, & Jennings, 2010; Tokic, 2010; Nouris, 2010).

Interval based techniques such as Upper Confidence Bound (UCB) base the level of exploration on the uncertainty of the estimate (Auer & Ortner, 2010; Powell, 2011). ϵ-greedy specifies the probability, ϵ, of taking an exploratory action explicitly, making it intuitive to most practitioners. Boltzmann exploration uses the estimated value of each state-action pair, $Q(s, a)$, and a scaling parameter,τ, to develop a distribution over the actions available from a given state, $a \in \pi(s)$. In contrast, the general idea of interval estimation is that those estimates with larger intervals will be sampled more often ensuring that as $n \to \infty$, $\sigma \to 0$ for a stationary environment (Powell, 2011; Auer & Ortner, 2010; Szepesvari, 2010). Techniques for decaying the ratio of exploration and exploitation, ϵ and τ, are discussed in the literature to ensure greedy behavior as $n \to \infty$, but for practical RL problems in non-stationary environments Sutton and Barto recommend the use of an appropriately "tuned" fixed ratio to avoid the inability to adapt to a changing environment (Powell, 2011; Sutton & Barto, 1998).

a. ϵ-Greedy

It is common to see a mixed strategy employed in the literature. One of the simplest of these techniques is ϵ-greedy, where an exploration rate specifying the probability of selecting a non-greedy action, ϵ, is specified in advance.

This strategy is popular for its simplicity and effectiveness (Powell, 2011; Russell & Norvig, 2010; Sutton & Barto, 1998). These methods choose a suboptimal step at random throughout the course of the learning period in conjunction with the specified ϵ. This guarantees that as the number of action selection opportunities goes to infinity, that each action will be sampled an infinite number of times satisfying conditions for convergence for some classes of value functions, but with no guarantees in the applied case since infinity is a long time (Russell & Norvig, 2010; Sutton & Barto, 1998).

Several derivatives of this approach exist in the literature such as ϵ-first and ϵ-decreasing. ϵ-first simply places the fraction of the episode, of length T, that is intended to be exploratory at the beginning of the episode. So, the policy is dependent on the current step with random action chosen prior to ϵT after which the greedy policy is followed. The

general idea with ϵ-decreasing is to allow ϵ to decrease with the number of iterations. Versions of this policy exist that leverage the number of steps in an episode and the number of visits to a state in order to adjust the policy. Let $N^n(s)$ be the number of times an agent has visited state s by iteration n,

$$\epsilon^n(s) = \frac{c}{N^n(s)}, \tag{12}$$

where $0 < c < 1$ is our initialization constant. So an exploratory action is chosen with probability $\epsilon^n(s)$ and the exploratory policy chooses a with probability $\frac{1}{|A|}$ and the probability of the policy selecting action a in state s, $P^n(s, a)$, is at least $\frac{\epsilon^n(s)}{|A|}$. The result of this approach is that each state will still be visited infinitely often even as the level of exploration decreases since (Powell, 2011),

$$\sum_{n=1}^{\infty} P^n(s, a) = \frac{\sum_{n=1}^{\infty} \epsilon^n(s)}{|A|} = \infty. \tag{13}$$

Multiple variations of this approach are found in the literature that employ similar strategies to address the challenge of adjusting the ratio of exploration and exploitation (Powell, 2011; Sato & Kobayashi, 2000; Sorg, Singh, & Lewis, 2010; Sutton & Barto, 1998; Tran-Thanh et al., 2010). ϵ-least-taken for example,

$$\epsilon = 1 - \min(\epsilon^a_{n,t}, 1), \tag{14}$$

where,

$$\epsilon^a_{n,t} = \frac{4}{4 + n^2}, \tag{15}$$

and n is the number of times action a was selected prior to trial t (Sato & Kobayashi, 2000). One potential disadvantage of ϵ approaches is that the random actions are selected with no regard to the information that has been obtained through previous sampling (Kaelbling et al., 1996).

Value difference based exploration (VDBE) is a another recent approach,

$$\epsilon(s_t, a_t, \sigma) = \frac{1 - \exp \frac{-|Q_{t+1}(s,a) - Q_t(s,a)|}{\sigma}}{1 - \exp \frac{-|Q_{t+1}(s,a) + Q_t(s,a)|}{\sigma}}, \tag{16}$$

where σ is a positive constant called the inverse sensitivity that must be tuned for each environment and task (Tokic, 2010). Note, however, that the fraction of the time to explore is reported as more intuitive to set than the scaling parameter, the temperature, employed by Boltzmann exploration techniques which make use of the information available regarding the value of state-action pairs to inform the level of exploration and exploitation (Sutton & Barto, 1998).

b. Boltzmann Exploration

Rather than simply choose an action at random Boltzmann exploration, alternatively referred to as a softmax action selection rule, makes use of the estimated value of the action, making the probability of choosing an action proportional to the value of its estimated value (Powell, 2011; Sutton & Barto, 1998). In our use, we adapt Boltzmann as shown below to overcome numerical precision error as the estimated value of each state-action pair divided by the scaling parameter goes to infinity, $\frac{Q(s,a)}{\tau} \rightarrow \infty$, an issue also identified by Hasselt as well (Hasselt, 2010).

$$P(a_i|\pi(s)) = \frac{\exp^{\frac{Q(s,a_i) - Q_{max}(s,a)}{\tau}}}{\sum_i \exp^{\frac{Q(s,a_i) - Q_{max}(s,a)}{\tau}}}, \tag{17}$$

where τ is a parameter known as the temperature. This parameter serves as a scaling parameter making the probability of selecting a greedy action go toward 1 as $\tau \rightarrow 0$ and producing a more exploratory sampling policy with larger values of τ. As with the ϵ-greedy techniques the level of exploration is often decayed over the course of the episode. Similar strategies are employed to achieve this effect.

In the following section, we will discuss two techniques that leverage information regarding the statistical quality of that estimate to determine the appropriate

exploration policy, interval estimation and upper confidence bound. For detailed comparison of Boltzmann and ϵ-greedy exploration that discusses when the techniques should be equivalent, see Appendix B.

c. Interval Estimation and Upper Confidence Bound

Methods from statistical sampling and experimental design have been applied successfully to this area. Interval based techniques, such as interval estimation, use measures of the uncertainty associated with an estimate, such as the variance or standard deviation, to determine the exploration policy (Kaelbling et al., 1996; Powell, 2011). The interval estimation algorithm, attributed to Kaelbling, takes this approach. The algorithm treats the action selection problem by maintaining an estimate of the expected value of an action as well as the standard deviation. The policy is,

$$\pi(s) = \max_a(\theta + z_\alpha \sigma_a^n) \tag{18}$$

where σ_a^n is the estimate of the standard deviation of $Q(s, a)$ and z_α is a parameter that determines the size of the interval. The general idea being that as the number of observations of a (s, a) goes to infinity the standard deviation goes to zero, for a stationary environment. (Strehl & Littman, 2005, 2004). The upper confidence bound sampling algorithm (UCB) takes a similar approach, using a different scaling factor to apply to the standard deviation.

$$\pi(s) = \arg\max_a\{\theta + C^{max}\sqrt{\frac{2\ln n}{N(s,a)}}\} \tag{19}$$

where C^{max} is the maximum possible contribution, but in practice an estimate of the 95th percentile for is used, and $N(s, a)$ is the number of times action a has been sampled (Powell, 2011).

d. Other Recent Approaches

Peeters et al. describe a learning automata approach to dynamically adjust the temperature on a Boltzmann function based on the action probabilities of multiple

agents (Peeters, Könönen, Verbeeck, & Nowé, 2008). Still and Precup provide an information theoretic approach to the exploration and exploitation problem, suggesting two approaches that incorporate the prediction accuracy (Still & Precup, 2012). Scott approaches the problem from a Bayesian viewpoint, using observation and MC methods to provide a parameter free method for determining when to explore and when to exploit.

Rather than adjust the policy-based on the variance of the estimate, Moody accounts for uncertainty by adjusting the value estimate itself (Moody & Saffell, 2001; Moody, Liu, Saffell, & Youn, 2004). He penalizes the state-action value estimate using techniques based on Sharpe's ratio,

$$S_t(s, a) = \frac{Q_t(s, a)}{\sigma_t^{Q(s,a)}}, \tag{20}$$

where the denominator represents the standard deviation of the long-term estimate of the value of a given (s, a) at time t, and the Sterling ratio,

$$S_t(s, a) = \frac{Q_t(s, a)}{\delta_t^{max} Q(s, a)}, \tag{21}$$

where the denominator represents the maximum negative change in the value of the long-term estimate observed up to the current time (Moody & Saffell, 2001; Moody et al., 2004). Using the Sterling ratio as a basis, he develops his downside deviation ratio (DDR), which penalizes the value estimate for downside variation, rewarding large positive returns and penalizing negative returns only, since his primary application for this technique is stock trading, where upside variance is often desireable.

$$DDR_t = \frac{Q_t(s, a)}{\frac{1}{t}\sum_{i=1}^{t} \min(Q_t(s, a), 0)^2} \tag{22}$$

This approach accounted for uncertainty by biasing the estimator against poorly performing state-action pairs, introducing estimation error for the value estimator, but positively impacting overall performance (Moody & Saffell, 2001; Moody et al., 2004).

3. Measures of Learning Performance

The value estimate calculated by the agent for each state-action pair provides the agent with a measure of goodness associated with each state-action pair and the accuracy of this approximation provides a means of comparing the relative performance of algorithms on similar tasks (Kaelbling et al., 1996). In addition to value approximation error, additional measures regarding the performance of algorithms and policies are eventual convergence to the optimal policy, speed of convergence, and regret. A discussion of these measures and their practical usefulness is described below.

a. Convergence

Several RL algorithms have been shown to be provably optimal in the limit, such as Q-learning (D. P. Bertsekas, 1995; Kaelbling et al., 1996; Powell, 2011; Watkins & Dayan, 1992). Convergence in the limit does not provide practical benefit in most real-world applications and this measure while relevant to the theoretical body of knowledge of the community, does not provide a ready means of evaluating an algorithms performance.

b. Speed of Convergence

This measure suffers from the same problem as convergence in the limit, since guarantees of optimality often only arise in the limit. Kaelbling suggests a metric such as the speed of convergence to near optimality, but concedes that this is also ill-defined and that measures based purely on speed of convergence might lead to unintended consequences in algorithm selection, such as incurring excessive penalties on the road to optimality (Kaelbling et al., 1996). On practical option often used to measure learning speed for sequential tasks in the literature is simply the mean number of times a goal state is achieved in a set period.

c. Regret

Regret provides a useful measure that captures the same information as measures related to convergence, but in a meaningful metric. Regret is defined as the expected

loss of reward or utility due to a deviation from the optimal policy (Audibert, Munos, & Szepesvári, 2007; Kaelbling et al., 1996; Kakade, Lobel, & Nazerzadeh, 2010; Kuleshov & Precup, 2010; Blum & Monsour, 2007).

$$L_T = T \max_a r(a) - \sum_{t=1}^{T} R_t, \tag{23}$$

where T is the total number of action selection opportunities in the episode and $r(a)$ is the reward received for action a. Kaelbling et al. cite a preference for this measure, but laments the fact that results documenting the regret of RL algorithms is not often available, though this has changed in more recent literature (Kaelbling et al., 1996; Blum & Monsour, 2007; Kocsis & Szepesvári, 2006; Tran-Thanh et al., 2010). Regret is used extensively as a measure of performance of algorithms used to solve n-arm bandit problems. For a Bayesian view of this problem see Appendix C.

d. Approximation Error

The accuracy of the value functions itself can provide insight into an algorithms performance as well especially when measured in conjunction with regret. The approximation error is the difference between the true value of a s, $V(s)$, and the approximation arrived at by RL algorithm, $\hat{V}(s)$. This measure can be examined for each s or, as suggested by Powell, as an aggregate across all s based on the frequency with which s is visited. Using his notation where $\nu^{(1)}$ is the error associated with the first algorithm to be compared, $p^{(1)}$, is the fraction of the time that the algorithm visited a (s, a) given algorithm one,

$$\nu^{(1)} = \sum_s p^{(1)} \| V(s) - \hat{V}^(1)(s) \|. \tag{24}$$

This measure treats all states, and could be applied to the estimates of state-action pairs, equally in the evaluation of the algorithms performance. Alternatively, based on the application, a weighting scheme might be appropriate tailoring error bounds based on the state-action pair's relevance to the achievement of the task goal (Powell, 2011).

e. Frequency of Optimal Action Selection

This measure provides an objective measure of performance bases simply on a ratio of the number of times the best action was selected for a given state and its complement (Kuleshov & Precup, 2010). This metric can be shown over-time or treated as an episodic measure. This is independent of the reward structure and the discount parameter often used in the estimation of the long-term expected reward of a state-action pair. During a learning session one would expect that this value would start out fairly low and would move toward 1 as the learning period progressed, but is highly dependent on the choice of stochastic policy and the ratio of exploration and exploitation. Exploration and exploitation is just one consideration in placing RL into a more human-like setting. Cognitive architectures provide a potential framework to allow the incorporation of RL methods into a model of the human information processing system.

B. COGNITIVE ARCHITECTURES

Cognitive architectures show great potential, but several limitations to their use in modeling and simulation applications currently exist. The first of these limitations is the lack of consistent benchmark environments and methodologies. This poses problems for validation efforts, suggesting that a context or use case-based approach might be more appropriate (Zacharias et al., 2008). A second limitation is the time required to develop and populate cognitive architectures. This relates to the third limitation, a limited ability to learn. New approaches to learning are required to overcome this brittleness problem encountered when architectures reach the limits of their domain knowledge. Another consideration, rather than limitation, is identifying the correct level of abstraction when representing human cognition for each use case. This is important since the ability to represent fine-grained detail within the information processing system exists, but the technology to verify and validate these structures does not. Future work is required in the following areas: tools to facilitate architecture development and instantiation, methodologies to facilitate knowledge base development and enhance explainability of developed models, meth-

ods for modeling groups, identifying the appropriate use of learning, and use-case based validation (Zacharias et al., 2008). This section will provide an overview of the general information-processing model of cognition, issues with cognitive modeling, and several of the more prominent cognitive architectures currently in use.

1. Considerations in Cognitive Modeling

Human cognition can be conceptualized using the information processing model formulation, which serves to enable experimentation and analysis as well as hypothesis generation (Wickens & Hollands, 2000). It is important to note that this model does not strive to identify where in the brain processes take place, but simply acknowledges that these processes must occur somewhere, leaving it to others to conduct empirical studies for the purpose of identifying which sections of the brain are involved with the different processes required by the general information processing model (Anderson, 2005; Anderson & Schunn, 2005). The model in Figure 5 is representative of the information processing model as related by Wickens and generally accepted in the human factors and cognitive science communities (Wickens & Hollands, 2000). Note that the software agent framework from Russell and Norvig mirrors this functionality at a more abstract-level. This should not be surprising, since artificial intelligence seeks to develop more capable software agents by emulating and understanding human behavior (Russell & Norvig, 2010).

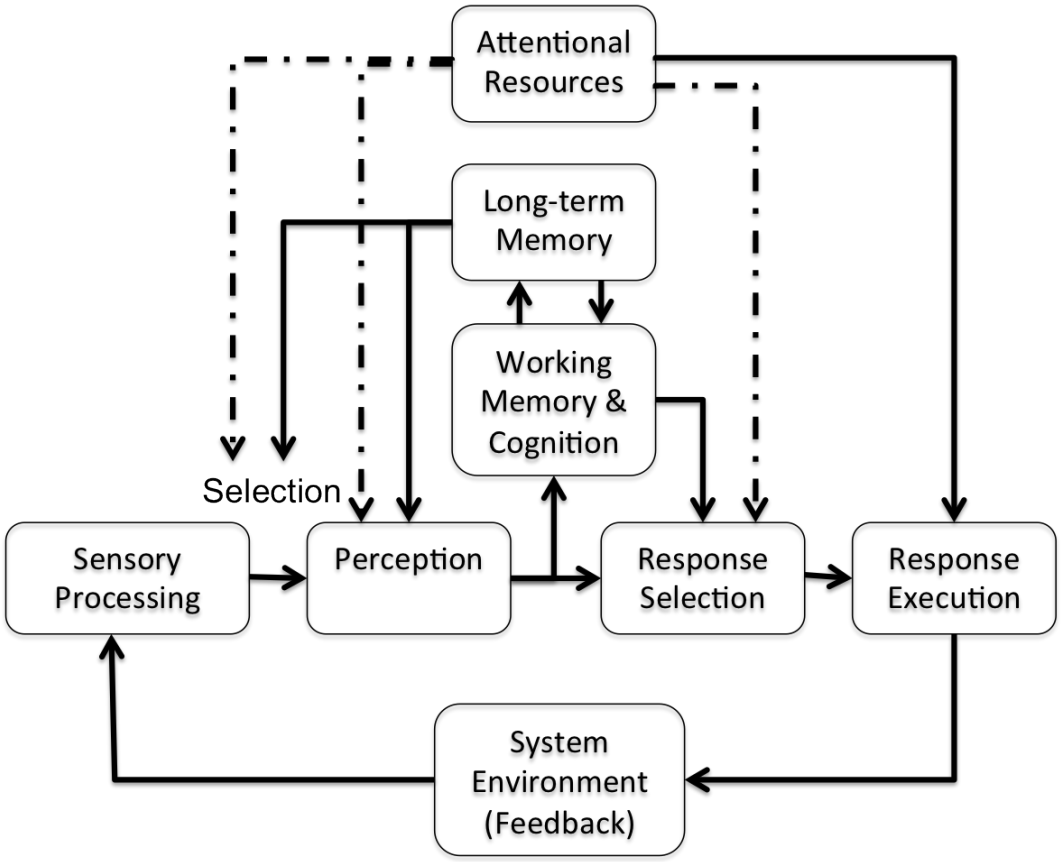


Figure 5: Information processing model of human cognition.

Information from the environment received by the sensory receptors enters into perception through sensory processing. Selective attention assists an observer in identifying which cues from the environment are relevant for processing given the current task and situation. Perception serves to provide meaning to the information received from the sensors, sometimes referred to as sense-making.

Two important characteristics of perceptual processing are that it is generally thought of as automatic and that it is driven by both bottom-up processing of sensory inputs and top-down processing of information from long-term memory regarding the current set of percepts from the environment (or situation) and expectations based on past experience.

Wickens distinguishes between perception and cognition based on the processing time associated with each, stating:

> cognitive operations generally require greater time, mental effort or attention. (Wickens & Hollands, 2000)

Cognition typically involves operations that utilize working memory. Working memory is characterized as a finite resource, that is highly vulnerable to disruptions based in attention. Once information has been rehearsed above a threshold it moves into long-term memory, to be retrieved as needed based on associations with other information from the current perceived state. Note the similarity with RL, whose main objective is to learn to map states to actions through a similar mechanism of association. Wickens provides only a brief treatment of response selection and execution, generally stating that an action from the potential actions is selected and in essence scheduled for performance just as in a discrete event system. The feedback loop is intended to represent two points (Wickens & Hollands, 2000):

- Flow of information can begin at any point in the system.
- The feedback loop is near continuous.

This is also a brief allusion to perceptual control theory a complementary view that makes the case that humans are control systems that rely on perception to maintain control of the system and that behavior can be managed by controlling ones perception (Powers & Treval, 1973). The attention component shows the role of attention in the process. The allocation of this finite resource drives the formation of perception and response selection, impacting the entire feedback loop. Attention interacts with long-term memory, making use of prior experience to allocate attention. This use of prior experience to learn what elements of the current perceived state are important again calls to mind concepts from reinforcement learning. The concept of divided attention impacts perception significantly. Operators attending to multiple tasks in an uncertain environment must divide their attention between

cues supporting each of the tasks. This divided attention degrades their ability to correctly attend to either task. This is a prime example of the quantity versus quality trade-off in terms of task performance and highlights the relative conceptual importance of attention as a means of influencing performance and makes a strong argument against multitasking.

The human information processing framework serves as a source of inspiration for placing realistic constraints on simulation based representations of human cognition. Taatgen and Anderson identify several constraints on cognitive modeling from the perspective of trying to replicate human cognition (N. A. Taatgen & Anderson, 2008). These constraints are proposed as considerations when developing cognitive architectures intended to replicate human cognition and serve as useful check points and points of comparison between architectures.

The first of these is working memory capacity, which serves the purpose of maintaining a representation of the current task environment and whose limitation is supported by Millers (1956) empirical work as referenced by Taatgen and Anderson. The authors describe the working memory constraint as one on cognitive function as opposed to a simple limiter on capacity. They further describe working memory as a relevance filter on perception, ensuring that the cognitive system is focused on relevant information (N. A. Taatgen & Anderson, 2008).

The second constraint on cognitive modeling is the serial bottleneck (N. A. Taatgen & Anderson, 2008). A debate exists regarding the presence of a bottleneck in central processing in human cognition centered around the allocation of attention to multiple streams of processing. It is thought that capacity constraints exist within peripheral processes, but that a more significant constraint exists in central cognition (Anderson, 2005; Anderson & Schunn, 2005). A common example used to illustrate this is the fact that humans can process multiple streams of perception and execute multiple simultaneous tasks, but cannot think about two things simultaneously Broadbent(1958) (Anderson, 2005; Anderson & Schunn, 2005). Through practice, however, humans have been shown to reduce the need for central cognition in select low-level cognitive tasks by developing a level of automatic-

ity that enables some level of parallel processing which is thought to assist in overcoming the constraint in central processing (Anderson, 2005; Anderson & Schunn, 2005).

Perception and motor system performance can be informed in great detail from empirical work, at least in regard to performance measures such as response time and signal detection, but the problem of informing learning from empirical work is considered more challenging. Taatgen and Anderson consider a model more constrained if it is required to learn its own knowledge, from either direct instruction or feedback, as opposed to having the knowledge specified by the modeler. This is particularly relevant for RL, where the entire task is to learn from interaction with the environment. This highlights a common technique employed to speed RL, the incorporation of domain knowledge. Taatgen and Anderson also state that learning can occur through examination of the environment and the identification of patterns, very much in line with RL and Klein's recognition-primed-decision making (N. A. Taatgen & Anderson, 2008; Klein, 1993).

Neuroscience provides two additional constraints that arise in discussion of cognitive architectures: constraints at the level of individual brain cells and constraints at the global brain architecture level. Two views of the impact of individual brain cells appear in the literature. The first view, reductionism, is that the characteristics of individual brain cells are not required to develop an understanding of human cognition (N. A. Taatgen & Anderson, 2008). The second view is that the neurons serve as important constraints on cognitive architectures, but this view raises the binding problem and the catastrophic interference problem (N. A. Taatgen & Anderson, 2008). The binding problem is how cognitive systems group features, or variables together. From an RL point of view, this speaks to the learning of the values of state-action pairs. The catastrophic interference problem refers to the fact that previously learned knowledge can be unlearned.

Cognitive architectures can now be informed by the results of work from neuroscience that identifies regions of the brain associated with certain functions. One of the most interesting open questions regarding cognitive architectures is the level of detail necessary to achieve generally intelligent agents (Laird & Wray III, 2010; Laird, 2008). From

a practical applied standpoint this question is really, what level of detail is required for the use case, since as Box says, "All models are wrong, some are useful."

2. Recognition-Primed Decision Making and Goals, Operators, Methods, Selectors

Recognition-primed decision making (RPD) developed out of the study of naturalistic decision making (NDM). NDM researchers seek to understand how individuals build expertise and apply it to decision making. In this paradigm, expertise is defined by explicit knowledge, in the form of facts and rules, and the more valuable tacit knowledge, such as the ability to recognize patterns, make subjective judgements, and make use of mental models, see Figure 6. Explicit knowledge is well represented in by first-order logic type rules. Tacit knowledge gained over-time by experience with a given situation or state of the world. Recognition of a given state that had been experienced before and an understanding of what actions are appropriate in that state are at the heart of RPD. The recognition-primed decision model describes how individuals match pattern to recognize a situation and depending on their expertise with that situation either leverage that experience to select an action or with less positive experience in a given situation they might mentally simulation potential outcomes of action alternatives (Klein, 1993, 2008). So, from this viewpoint, RL would be used to build tacit knowledge and as the RL algorithm converged this would lead to a transformation of those rules into explicit knowledge.

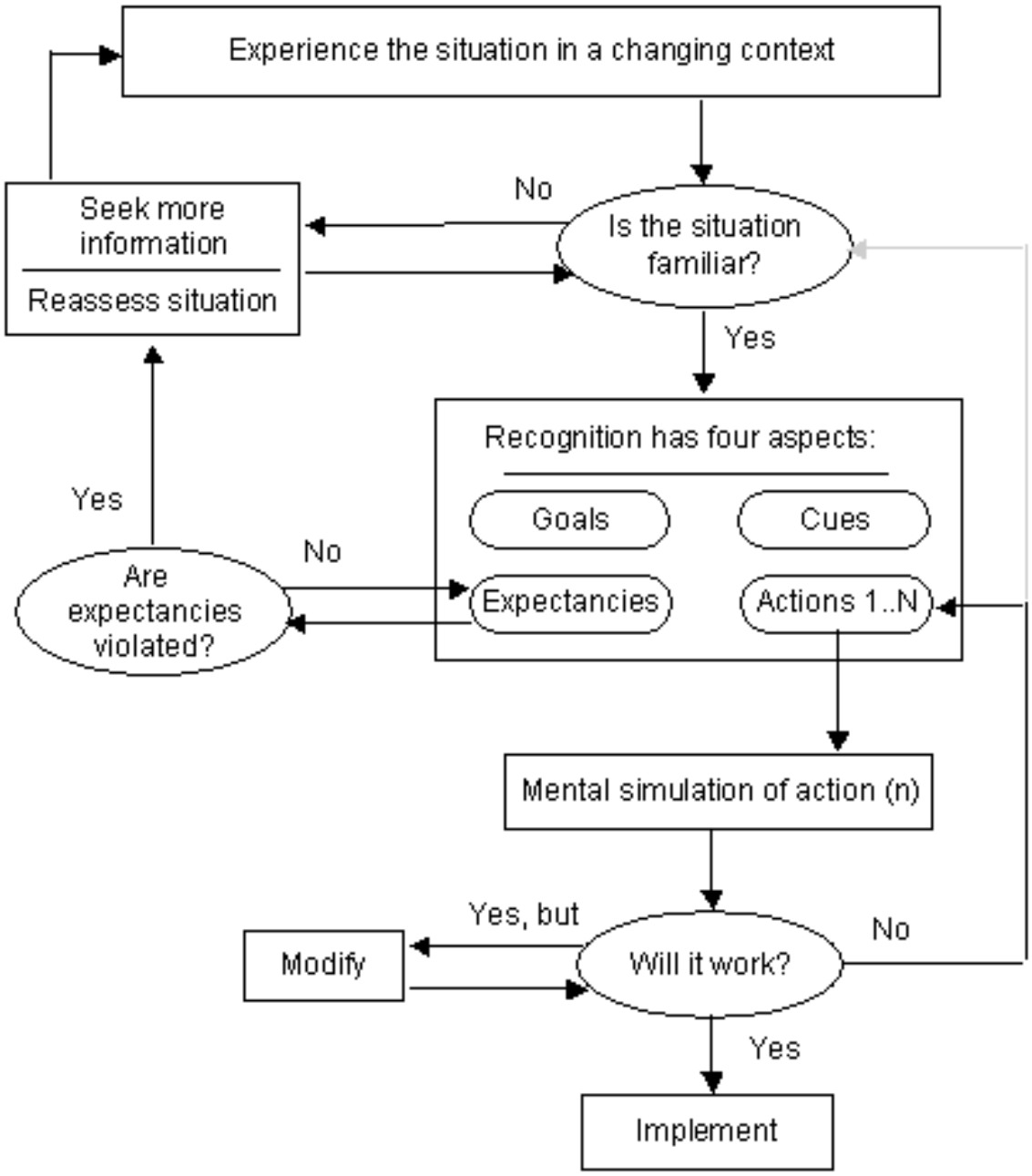


Figure 6: Recognition primed decision making (Klein, 1993).

Goals, Operators, Methods, and Selection (GOMS) task analysis provides a useful framework for analyzing and representing goal-directed behavior cite Card,Moran, Newell

(John & Kieras, 1996; Olson & Olson, 1990; Oyewole & Haight, 2011; Craig et al., 2012). GOMS derived first order logic rules are a form of cognitive task analysis that reduces human interaction with a computer down to first principles. We can also use GOMS frameworks to drive autonomous behavior. In this sense, the goal is the objective the agent is seeking to obtain, the operators are actions that can be performed to obtain the goal, methods are sequences of operators that accomplish a goal, and selection rules adjudicate conflict resolutions between methods. This type of framework has been implemented effectively in the form of first-order logic rules in productions systems, such as CLIPS. Many of the existing cognitive architectures leverage production systems and GOMS like rules to represent domain knowledge and drive behavior (N. A. Taatgen & Anderson, 2008; Sun, 2007b; Laird & Wray III, 2010).

3. Review of Cognitive Architectures

This section will review three of the more prevalent cognitive architectures in broad use: Atomic Components of Thought (ACT-R), State, Operator, and Results (SOAR), and Connectionist Learning with Adaptive Rule Induction On-line (CLARION).

a. Active Components of Thought

ACT-R, whose development started in 1983, has traditionally focused on serving as a platform for research on cognition and representation of fundamental psychological processes, see Figure 7 (Zacharias et al., 2008). ACT-R uses a combined form of symbolic and numerical representation with production rules firing based on log odds of success of a particular rule in a given situation, its activation level (Sun, 2007b). ACT-R deals with the notion of working memory capacity through its use of declarative memory which decays as the size of the information pushed in increases. In this manner, ACT-R constrains the agent's information processing ability based on a capacity based representation of working memory (N. A. Taatgen & Anderson, 2008). ACT-R has been used in a number of applied settings to include the modeling of adversarial behavior (Zacharias et al., 2008).

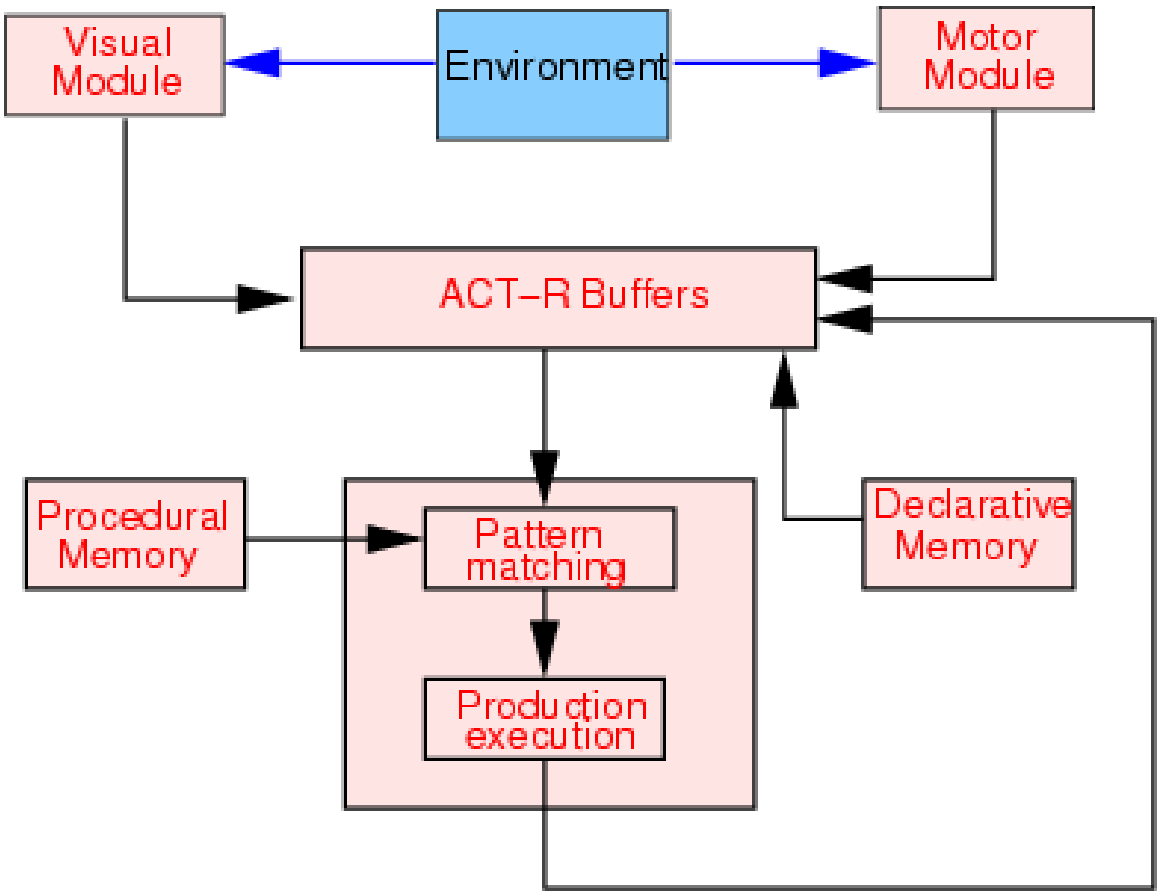


Figure 7: ACT-R top-level conceptual diagram.

ACT-R imposes the serial bottleneck in central processing by allowing only one rule to fire at a time in its central production system and by limiting the number of cognitive steps that can be taken in a given decision cycle (N. A. Taatgen & Anderson, 2008). ACT-R further constrains its production system by limiting matches to items currently in its buffers implying that it cannot match against all items in its declarative memory and must instead retrieve and examine one item at a time. Learning occurs in ACT-R by combining rules over-time into new sets of behavior that are commonly associated with each other, transforming general knowledge into task specific knowledge over-time (N. A. Taatgen & Anderson, 2008).

b. State Operator and Results

The SOAR architecture has gone through eight major versions between 1982 and 2007, all maintaining a pure symbolic processing approach and using production system rules for long-term memory (Laird & Wray III, 2010; Laird, 2008). The traditional SOAR, up through SOAR 8, used symbolic long-term memory, representing knowledge as a set of production system rules, and short-term memory, which maintains a current assessment of the situation based on perception of the current state informed by long-term memory.

SOAR is characterized as taking a functional approach to the representation of working memory capacity by Taatgen and Anderson, where limits on working memory serve to separate relevant from irrelevant information in the given task context (N. A. Taatgen & Anderson, 2008). The general processing cycle in SOAR is to receive an input from perception into short-term memory, update the agent's goal based on the new information, and evaluate and select operators to achieve the goal, much like a GOMS framework, see Figure 8. Fixed decision procedures then select the appropriate operator, with arbitrary conflict resolution mechanisms in place. The actions associated with the chosen operator are executed by the rule-based system and the output passed to the environment for execution (Laird & Wray III, 2010; Laird, 2008).

SOAR is capable of learning new rules from direct instruction has been used in a variety of applications ranging from expert systems to the control of autonomous agents (N. A. Taatgen & Anderson, 2008). SOAR has been used in military simulations to replace human role players and by the Institute for Creative Technologies (ICT) at the University of Southern California to control virtual characters for game-based training applications (Zacharias et al., 2008). Laird highlights extensions to the traditional SOAR in the latest version, SOAR 9, to provide capability for long-term memory representation, additional learning mechanism, and non-symbolic processing (Laird & Wray III, 2010; Laird, 2008; Wray & Jones, 2006).

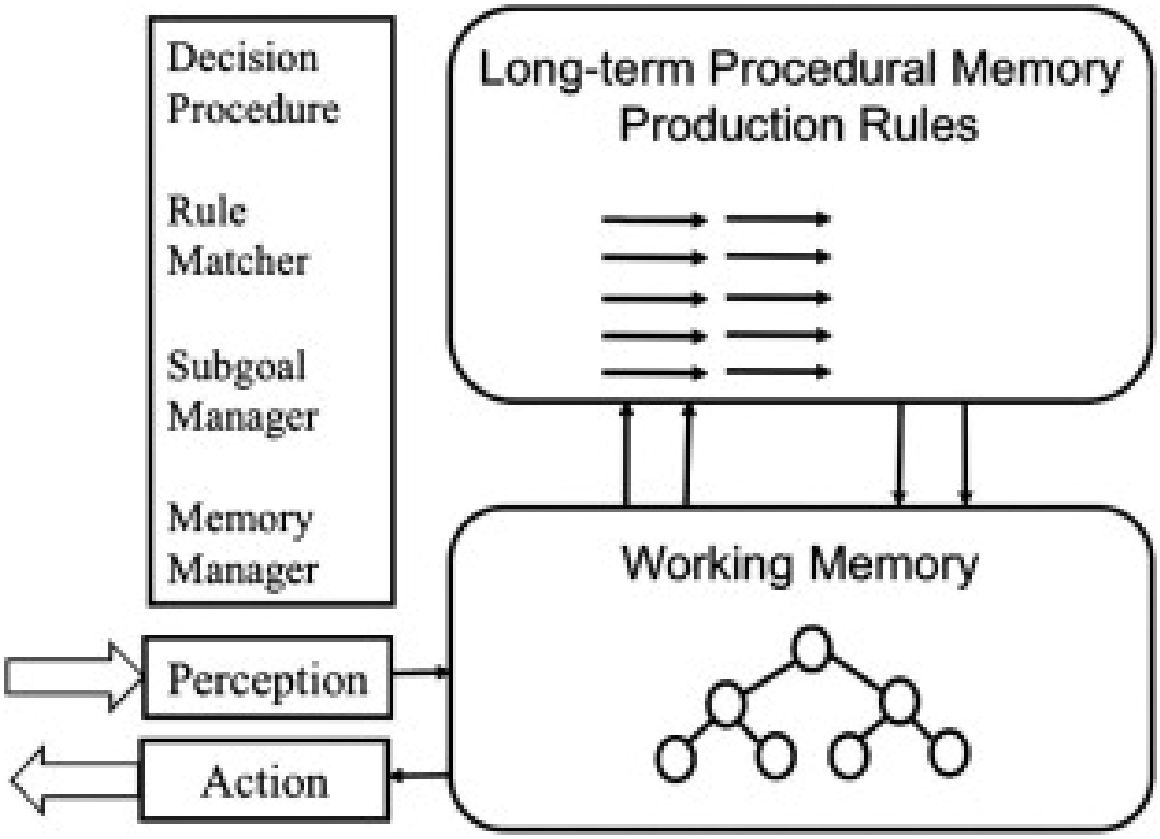


Figure 8: SOAR top-level conceptual diagram.

Laird states that the goal of SOAR 9 was to retain the strengths of the original SOAR while expanding SOAR to more closely resemble human capabilities (Laird & Wray III, 2010; Laird, 2008). The new version of SOAR incorporates a notion of activation into its representation of working memory, inspired by ACT-R. RL is integrated into the selection of operators using an ϵ-greedy algorithm. The concept of emotion is represented in the current version of SOAR using appraisal theory. Appraisals lead to emotions, eventually being expressed as an intensity of emotion, which is used in conjunction with RL as part of the reward structure. Declarative knowledge can be built up from information that first moves into working memory. Episodic memory, task independent information that can be used to aid in reasoning across tasks, contains memory of experience over-time

providing an idea of context. This is believed to be a key enabling capability for mental simulation and prediction, two functions believed to enable decision making under conditions of uncertainty.

c. Connectionist Learning with Adaptive Rule Induction On-line

CLARION is a cognitive architecture consisting of four subsystems: the action-centered subsystem, the non-action centered subsystem, the motivational subsystem, and the meta-cognitive subsystem, see Figure 9. Each subsystem provides two levels of knowledge representation, a top-level for explicit knowledge representation and a bottom level for implicit knowledge representation. These bottom-up associations between action, state and outcome inform action selection (N. A. Taatgen & Anderson, 2008). Interaction occurs between the two levels during action selection and learning. The action-centered subsystem controls all actions, external to the agent and internal. The non-action centered subsystem stores and maintains general knowledge. The motivational subsystem determines motivations for perception, action and cognition. The meta-cognitive subsystem controls the system of systems, providing central control (Sun, Zhang, & Mathews, 2006).

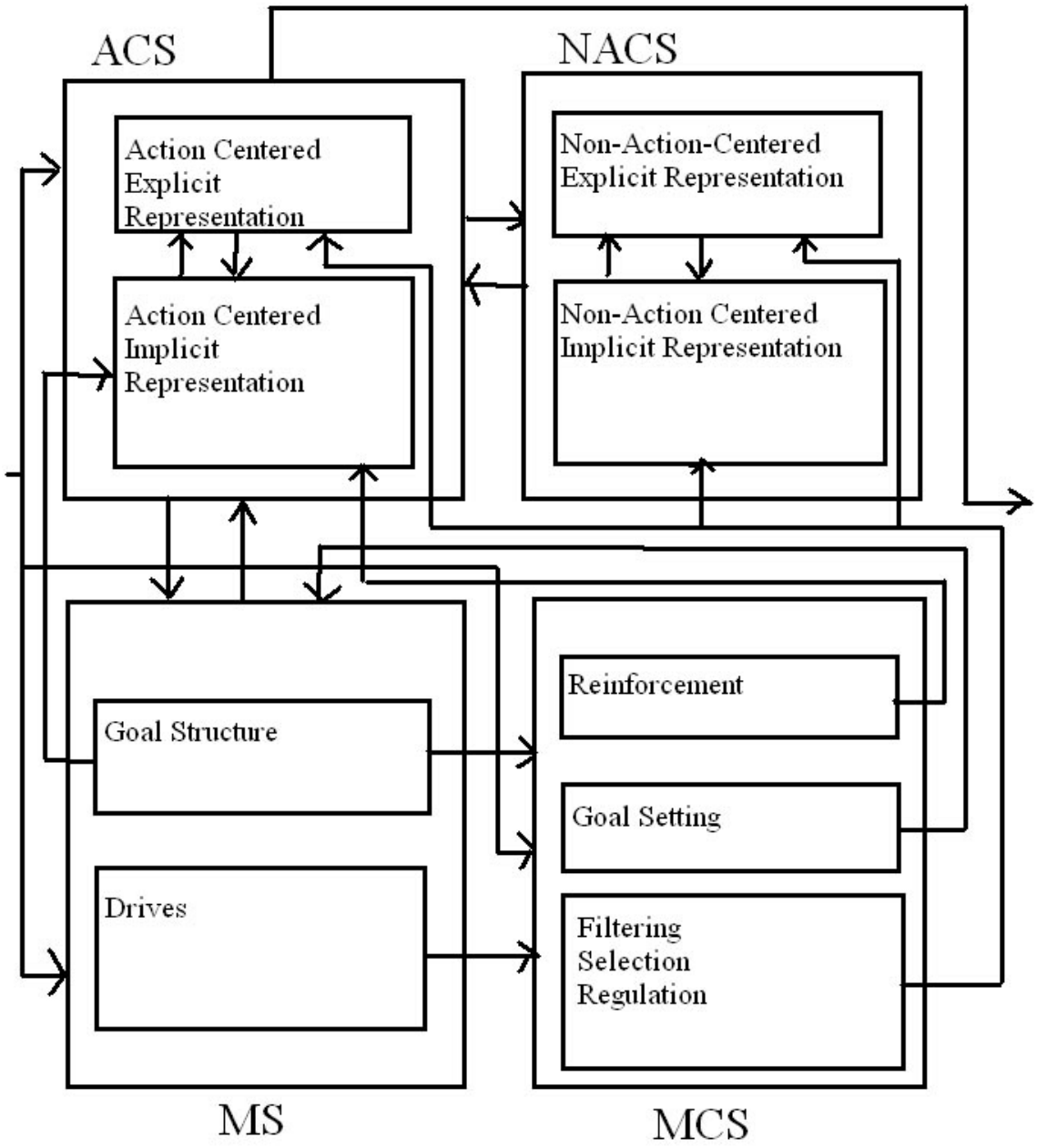


Figure 9: CLARION top-level conceptual diagram.

CLARION uses rule systems implemented in a neural network to constraiin the computational power of the system and focus on more local situations (N. A. Taatgen & Anderson, 2008). CLARION has mechanisms for implicit and explicit learning (N. A. Taatgen & Anderson, 2008). CLARION is described by Sun as a hybrid connectionist-symbolic system and has been applied to modeling organizational decision-making, with CLARION based agents interacting as part of an organization and in an analysis of survival strategies of tribal societies. The authors conclude with the need to explore the implications of individual cognition on political systems, justice and individual motivating factors (Sun & Naveh, 2007).

4. Learning in Cognitive Architectures

The degree of learning to include in agents depends on the purpose of the modeling effort and the level of sophistication in the agents. The general problem is to allow agents to learn which action choices lead to desirable outcomes within the agent environment. This has advantages over trying to script every possible state an agent might encounter, which will be inherently brittle. Learning in cognitive architectures intended for social simulation can be further characterized by the objective of modeling individual and group human behavior as realistically as possible. Multiple approaches to the problem exist with various advantages and disadvantages (Brenner, 2006; Sutton & Barto, 1998). The use of learning techniques facilitates the development of adaptive agents that update their behaviors based on information received over the course of a simulation run. This capability enables dynamic communications patterns, the changing of affiliations over-time, and behavior changes in response to situations in the environment. The use of learning agents can also potentially enable verification of the dynamics of the environment.

ACT-R implements learning in the determination of activation levels in its declarative memory. The activation level of an item in declarative memory is linked to the frequency of its use, with decays in activation levels being experienced over periods of inactivity, in line conceptually with reinforcement learning. The activation level represents the probability that the item in memory will be needed given a certain state of the envi-

ronment. Production rules retrieve these chunks from memory through a buffer, with the activation level linked to the retrieval time. Each production rule has an associated utility value, continually updated by learning algorithms, based on the notion of a estimated cost and probability of reaching the goal state given the selection of that rule. ACT-R also adds items to declarative memory and constructs new production rules, similar to SOAR, by combining associated rules, rules fired sequentially, into a single rule (N. Taatgen et al., 2006; Wray & Jones, 2006). ACT-R and SOAR contrast with CLARION in their approach to learning (Sun, 2007b).

CLARION combines a bottom up and top down approach to learning, with agents that use model free methods from RL, such as Q-learning, to determine actions choices from the bottom up, under the supervision of top down rules and chunks similar to those found in ACT-R or SOAR. The output of both bottom-up and top-down processes is an action choice. Feedback for these processes comes to the agent in the form of an internal and external signal. Internal feedback exists in the form of motivation, external feedback must be specified by the modeler. Learning and action choices reside in the action centered subsystem with goals being specified by the motivational subsystem and the meta-cognitive subsystem. The meta-cognitive subsystem provides the agent with information regarding its own internal state based on internal feedback from the motivational subsystem (Sun, 2007b). The use of motivation and its impact on goal setting is consistent with the updated view of Maslow's hierarchy of needs.

5. Agent Based Models, Cognitive Modeling, and Complexity

Agent-based modeling is defined by the National Research Council as the computational study of systems that are complex in the following sense: (1) the systems are composed of multiple interacting entities and (2) the systems exhibit emergent properties properties arising from entity interactions that cannot be deduced simply by averaging or summing the properties of the entities themselves (Zacharias et al., 2008). Multi-agent systems as defined by Shoham and Leyton-Brown are those systems that include multiple autonomous entities with either diverging information or diverging interests, or both

(Shoham & Leyton-Brown, 2009). Ferber refers to multi-agent systems as societies of agents that interact and coordinate their behaviors toward some common goal (Drogoul & Ferber, 1994; Ferber, Gutknecht, & Michel, 2004). Ferber goes further to distinguish agent centered multi-agent systems (ACMAS), which he refers to as the classical view of MAS, from what he characterizes as organization centered multi-agent systems (OCMAS). Where ACMAS was centered on cognitive states of the agents being modeled, OCMAS focuses instead on roles, groups, tasks and interaction protocols. Gilbert, describing agents from a MAS point of view attributed to Wooldridge and Jennings, states that agents "are processes implemented on a computer that have autonomy (they control their own actions); social ability (they interact with other agents through some kind of language); reactivity (they can perceive their environment and respond to it); and pro-activity (they are able to undertake goal-directed actions) (Gilbert, 2008). The relationship between agent based models and multi-agent systems is loosely defined at best. Both leverage similar concepts and techniques, so it appears that the primary difference rests in the use case.

Agent based modeling and multi-agent systems share in common the idea of intelligent agents that use some level of information from the environment in conjunction with rule sets or algorithms of varying complexity to select actions that allow them to interact with their environment and other agents. The line between these two areas is not distinct, with the clearest distinction possibly being formed by intended use and intended environment. From this perspective, their approach is consistent with the definition of an agent from the field of artificial intelligence as stated by Russell and Norvig as anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators (Russell & Norvig, 2010). Both fields focus on the use of software agents as members of complex systems.

The discipline of cognitive modeling seeks to represent human cognition within software agents for a variety of uses through the development and application of cognitive models and cognitive architectures (Zacharias et al., 2008). Cognitive architectures, described in some contexts as micro-level formal models, are simulation based models of

human information processing often built to emphasize distinct aspects of cognition, based on the use case. Agent based social simulations represent human cognition at varying levels of sophistication (Zacharias et al., 2008), but typically adhere to the most rudimentary level of an agent, a reflex agent, as defined by Russell and Norvig and reviewed previously. Summarized here, an agent senses information, or percepts from its environment, using sensors, updates its internal representation of the world, and selects actions based on this updated internal state (Russell & Norvig, 2010). Depending on the needed level of resolution the agent can represent either an individual or group of individuals, though the use of cognitive architectures in the representation of group cognition is not well developed (Zacharias et al., 2008). Sun points out that agent based social simulations and cognitive architectures have developed in relative isolation from each other, but that the use of appropriate cognitive architectures could benefit agent based social simulation by providing a realistic basis for the representation of individual agents. Sun further identifies cognitive social simulation as a path forward for the intentional combination of these two fields into cognitive social simulation (Sun et al., 2006). While the potential for agent based social simulations and cognitive architectures to enable a multi-level examination of human behavior including the sociological and psychological perspectives respectively exists (Sun, 2007b), the National Research Council is less clear on the use of cognitive architectures to represent group cognition (Zacharias et al., 2008).

A cognitive architecture provides the specification for those features of an intelligent agent or system, depending on the use case, that are invariant over-time and across applications (Langley et al., 2009). The development of cognitive architectures, fixed infrastructures to support the acquisition and use of knowledge, springs from the pursuit of generally intelligent entities as opposed to specialized algorithms. Generally intelligent entities are distinguished by their ability to be applied to a wide variety of tasks, using diverse knowledge and experience in complex environments (Laird, 2008). Cognitive architectures differ from cognitive models, which can be thought of as more narrowly scoped

micro models while cognitive architectures generally seek to implement a unified theory of cognition (Zacharias et al., 2008).

Sun defines a cognitive architecture broadly as a "domain generic computational model, capturing the essential structure and process of the individual mind (Sun et al., 2006). Taatgen and Anderson describe cognitive architectures as "on the one hand echoes of the original goal of creating an intelligent machine faithful to human intelligence, and on the other hand attempts at theoretical unification in the field of cognitive psychology (N. A. Taatgen & Anderson, 2008). Multiple cognitive architectures exist and have been applied to a variety of use cases, but the common components of a cognitive architecture, as identified by Langley et al. and Laird, include short and long-term memory, language to represent elements stored in memory, and functional processes that leverage these structures and is consistent with Wickens human information processing view (Laird, 2008; Wickens & Hollands, 2000). The National Research Council expands the list of cognitive and perceptual processes to include: attention, situational assessment, goal management, planning, meta-cognition, learning, action selection, and memory (Zacharias et al., 2008). Psychologically oriented cognitive architectures are important because they can facilitate the understanding of human cognition, can inform understanding of societal-level collective behavior, and are relatively realistic and human like (Sun, 2007b). Cognitive architectures within social simulation serve to add structure and specificity to the key component of these models, the individual actors responses that provide the link between the micro and macro-levels.

Agent based models and multi-agent systems focus on the use of intelligent agents within simulation for a variety of purposes. The level of sophistication the agents employ varies by use case in these settings as does the need to represent human like cognition and behavior. Cognitive architectures focus on the representation of human cognition and behavior by seeking to replicate the functions of the human information processing system. Sun proposes the creation of cognitive social simulations, which combine these disciplines allowing for more detailed representation of human behavior at the micro-level upon which

agent based modelings macro-level analysis rests (Sun et al., 2006). The efficacy of this approach has yet to be fully explored in the literature, but holds promise as a means to provide sophisticated behavior within agents while maintaining traceability to conceptual models required for validation. The link between the micro and macro system behavior level will be expanded below.

The study of complex social systems go back hundreds of years with Adam Smiths The Wealth of Nations (1776) being one of the earliest discussions on the topic (Smith, 1966). Smiths invisible hand guides self-interested rational agents into well formed market structures (Miller & Page, 2007). The traditional tools of social science often constrained the modeling of social systems. Tools developed in the study of complex systems, such as agent based modeling, have been identified and come into use by researchers studying complex social systems (Tesfatsion & Judd, 2006). The relation between complex systems research and complex social systems is especially relevant to understanding the transition of a society from a relatively unstable state, such as during an open insurgency, to the more stable post conflict state, with an inactive or non-existent insurgency. Understanding the perception, motivations, and intention of the diverse actors at play in a conflict ecosystem are essential to gaining an understanding of how the social system can be transitioned from the undesirable chaotic state of insurgency, to the more stable and desirable post conflict state (Mansoor, 2007). Stable, in this context, is certainly relative to the given social system and its concept of normal.

Miller and Page state that complexity arises in a system when dependencies among the elements of the system become important to the normal behavior of the system to such an extent that the removal of one element from the system alters the systems behavior to an extent much greater that intuitively expected given the relative perceived contribution of the removed element (Miller & Page, 2007). These complex systems provide challenges to those seeking to understand their responses under varying conditions. Social systems tend toward complexity, with multiple intelligent social agents interacting through a variety of processes and connections. Systems such as these are non-linear and do not lend themselves

to analysis by decomposition, making holistic approaches more appropriate for system-level analysis (Miller & Page, 2007).

The characteristics that make agent based simulation an attractive alternative for the analysis of complex systems make them a viable alternative for use in examining complex social systems. The application of agent based models to this domain also suffers from the same limitations to their application within the field of complex systems. In addition to these limitations, the use of these tools in the representation of social systems requires the representation, at some level, of human behavior. A lynchpin in the future validation of cognitive social simulation, at the macro-level, is the validation of the cognitive architecture used, at the micro-level.

The connection between agent-based models and the social sciences is illustrated by the field of economics, which models economic behavior as the result of rational actors, with ties to models based on individual actors from its inception. Epstein highlights this relationship, pointing out that certain social systems are distributed, asynchronous, decentralized and dynamic, such as trade networks (Epstein, 2006). When market prices reach equilibrium the society has executed a calculation and converged on a solution. Convergence to social norms is equivalent to convergence to strategy distributions in n-person games or to an optimal policy when mapping state-action pairs in reinforcement learning (Epstein, 2006). The first applications of simulation to the social sciences coincided with the first uses of computers in university research settings in the 1960s (Gilbert, 2008). Many of these early efforts focused on modeling for prediction rather than understanding-typically the goal of most social science research. While these early efforts met with mixed results, and served to generate healthy skepticism of simulation, an approach which gained traction was the idea of micro-simulation. Micro-simulation modeling represents individual members of a population, based on polling or other data sources, and follows them forward through time through a series of transition probabilities. This technique has been successfully applied by several European nations (Gilbert, 2008).

An interesting aspect of agent models that Epstein devotes some time to is their ability to allow multiple social theories, which he refers to as social computations, to be active at once in an experimental framework. Epstein, a proponent of the use of these models by social scientists, proposes the use of agent based social simulations to facilitate interdisciplinary social science work. In general, Epstein adheres to what he refers to as a generative approach to social science (Epstein, 2006). This approach relies on the use of agent based models to facilitate the representation of individual actors with simple rule sets, what he refers to as micro-level behaviors. These simple rule sets represent the simplest possible representation of agent cognition. This generative approach proceeds with experimentation with the agent based social simulation with micro-level specifications, seeking to find a set of such specifications that produces macro-level output data that matches some historical data set or observed social phenomena of interest.

This micro to macro-level mapping allows for the interaction of behavioral and social sciences to some degree. The successful identification of such a set of micro-level behaviors leaves the analyst or social scientist with a screened hypothesis that has not been rejected given the conditions of the experiment. This is a critical point to make regarding this approach. In order to accept that the micro-level behaviors constitute a potential explanatory hypothesis describing the system under study, the modeler also has to accept that the interaction rules of the system and the data that have been used to populate the model are accurate and valid. In some domains, this may be a minor requirement, but in the application of these techniques to human behavior this serves as a major limitation of the generative approach. This also highlights the need for the integration of individual and group cognitive architectures into agent based social simulation in a more robust manner. The use of psychologically oriented cognitive architectures at the individual-level in this domain can provide structured integration between Epsteins micro-level specification from the field of psychology and understanding of societal-level collective behavior (Sun, 2007b).

A needed area of research is the exploration of group cognitive architectures based on social psychology. Architectures based on social psychology would allow the researcher to change the level of analysis while preserving the benefits of the use of cognitive architectures as a formal specification. Cognitive architectures provide a detailed specification of the framework of human cognition, including realistic constraints, capabilities and tendencies based on cognitive processes, but possess validation concerns as well (Sun, 2006).

Epstein as well as Miller and Page describe the use of agent based social simulations to explore the generation of equilibrium conditions from societies of boundedly rational agents in decentralized locations (Epstein, 2006; Miller & Page, 2007). Gilbert adds to this highlighting the fact that societies are made up of individual actors with dynamic, nonlinear interactions where knowledge and materials are exchanged that impact the future behavior of the individual actors (Gilbert, 2004). The dynamic nature of the system makes it impossible to study the society as a whole by examining individual members in turn, since macro-level system behavior emerges from the interaction of individually motivated actors on a dynamic landscape (Gilbert, 2004). Sun proposes the use of cognitive social simulation to explore these interactions at multiple levels, with psychologically based cognitive architectures embedded in agents within social simulation (Sun, 2006). Borrill and Tesfatsion propose that agent based models provide a local observer view of the world as opposed to a Gods eye view, and that this local observer view provides great insight for the social sciences highlights the importance of the cognitive architecture specifications within social simulation. They provide two examples of the application of agent based models to the social sciences: one in the analysis of power consumption in the U.S. and the second in the field of information storage design (Borrill & Tesfatsion, 2010).

The application of agent based models to the social sciences has focused on the use of simple behaviors at the individual agent-level to generate macro-level social phenomena as described above. Common phenomena explored include segregation, trust formation, and resource constrained games (Schelling, 1972). The use of these models in conjunction with cognitive architectures has not been fully explored, but is required to fully represent

complex phenomena such as belief revision, changing affiliations, coercion and persuasion, and the formation of trust. The following section will discuss the topic of emergent behavior, a topic often used to refer to the micro to macro link, in greater detail (Zacharias et al., 2008).

Any discussion of agent based social simulations or agent based models will eventually touch on the topic of emergence (Zacharias et al., 2008). Emergence, the idea that aggregate behavior arises from the interaction of local individual behaviors, provides one of the more contentious points of discussion in the area of complex systems and complex social systems (Epstein, 2006; Gilbert, 2004). The description of emergence most often used describes emergence as the result of individual, localized behavior aggregated into a global behavior disconnected from its origins (Miller & Page, 2007). This implies that the macro-level system behavior is robust to changes in the micro-level system behavior as specified in what traditional agent based model research refers to as micro-level behavior and cognitive social simulation specifies in the cognitive architecture. Miller and Page point out that emergence can occur at many levels within the system of systems, providing natural breakpoints in some cases for distinguishing sub-systems that might not previously have been apparent. Epstein concurs with this assessment, by characterizing emergence as phenomena relative to the current state of knowledge describing the systems of interest (Epstein, 2006). In this view, the definition of emergent behavior for a system under study relates to those unexpected events or conditions that give rise from micro-level behavioral interactions. Sun refers to this as the micro-macro link and emphasizes the central role of cognitive architectures in this phenomenon (Sun, 2006). When emergent macro-level behaviors are observed that correspond to phenomena, social or otherwise, observed in the real-world system, a potential explanatory hypothesis at the micro-level has been identified (Epstein, 2006). Those that ascribe to this view are clear that this is not the end of the analysis, but serves to focus further empirical work with the system to accept or reject this new hypothesis resulting from the observed emergent behavior (Epstein, 2006).

Miller and Page use the Law of Large Numbers and the Central Limit Theorem as examples of theories that describe how individual behaviors generate aggregate patterns that are robust against variation within limits (Miller & Page, 2007). The authors attribute the effectiveness of these two theorems to the concept of disorganized complexity. The key feature for this discussion is that the interactions of local entities tend to average out as an increasing number of independent actors are added to the system. The system eventually reaches a point where system-level predictions are possible through the above stated theorems, but individual-level forecasts are still elusive. Gilbert concurs with this assessment of emergence in social systems using the example that while we can identify the mission of a political organization, the identity does not always transfer to the individuals that belong to the organization (Gilbert, 2004).

Miller and Page coin a term organized complexity in referring to those properties of complex systems that arise from the interaction of intentional agents in systems that provide feedback. In these types of systems, variations no longer average out, but rather become reinforcing causing the system to exhibit macro-level behaviors that would be termed emergent in this domain (Miller & Page, 2007). From this view point emergent macro behavior could be said to arise in systems with organized complexity as defined by Miller and Page, resulting from the central role of the cognitive architecture specification in the case of cognitive social simulation. This also meets the definition of emergence from Gilbert which defines emergence as any phenomena arising from a system that requires new categories in its description that are not necessary for the description of the system components (Gilbert, 2008). Going back to some of their earliest uses of social simulation, Schelling used agent based models to study tipping points, particularly in regard to racial residential segregation processes, providing exemplars of this micro to macro link (Schelling, 1972).

Gilbert provides several examples of what he terms emergent behavior from real social systems. He uses as examples such phenomena such as segregation into neighborhoods, the evolution of language, and migration. He points out that in social systems the actors can actually observe and identify emergent behavior and in some cases adjust their

behavior with what they perceive as the emerging trend, highlighting the importance of cognitive architectures (Gilbert, 2004). This corresponds to Miller and Pages view of systems that exhibit organized complexity, with micro-level behaviors serving to reinforce emerging macro-level trends (Miller & Page, 2007). These cognitively aware actors reason and make choices at the individual-level, with results apparent at the macro-level. Castelfranchi, quoted by Sun, sums up the problem nicely: The real problem is modeling how we play our social roles, while being unaware of the functional effects of our actions, not only with our routine actions but even when doing something deliberately for our own subjective motives (Sun, 2006).

Cognitive architectures play a central role in the generation of emergent behavior in complex social systems (Sun, 2006). Emergent behavior at the macro-level stems from the interaction of micro-level behaviors at the individual agent-level, as described above. Psychologically based cognitive architectures provide structure that encapsulates the cognitive functions of the agent and can lead to cognitive emergence, or bottom up learning, as described by Castelfranchi (2001) and referenced by Sun (Sun, 2006). At a coarse-level, these functions include perception, long and short term memory, and action selection. Since it is the resulting action selection from each individual that in the aggregate leads to emergent behavior, cognitive architectures, the action selection component in particular, implicitly influence macro-level behavior generation at the societal-level. The use of cognitive architectures provides structure to the representation of human behavior and requires detailed specification, enabling greater documentation and potentially aiding in the exploration of the micro to macro link and validation efforts for this class of models (Sun, 2006). The need to represent micro-level behaviors in order to gain an understanding of the system under study, rather than the individual actors, leads to a discussion of the need for a holistic approach to system analysis to inform a holistic analysis.

Agent based models are tools for holistic analysis of systems, but require a reductionist approach in the development of micro-level behaviors, a minimal cognitive architecture, for individual actors. Agents are intended to represent human behavior in the most

simplified manner that is still useful (Gilbert, 2008). The application of the reductionist approach to agent based modeling simply assists in the identification of a logical starting point for what Epstein calls the generative approach (Epstein, 2006). Discussion of the reasons that a pure reductionist approach to complex social systems is inappropriate is provided by Miller and Page and Epstein at some length (Epstein, 2006; Miller & Page, 2007). The arguments presented by both groups tend to center around the existence of emergent behavior in complex systems that cannot be easily explained by simple system decomposition as was discussed in the previous section. The authors use the partitioning of economics into micro and macro economics and the partition between the behavioral and social sciences as examples from well studied fields that recognize that the reductionist view does not fully account for system-level behaviors (Epstein, 2006; Gilbert, 2008; Miller & Page, 2007).

Agent based models provide a computational tool for use in creating a holistic view of the system under study. Computational tools do not differ in the evaluation of their utility from theoretical tools or mathematical tools they should be judged on their ability to enhance science and simplify a task. Theories by contrast should be judged on their ability to improve understanding of some phenomena of interest (Miller & Page, 2007). Computational tools force the modeler to be precise, whereas theories expressed in natural language can be open for interpretation (Gilbert, 2008). When modeling the system under study from this view point, the reductionist method of decomposition of the system to its constituent parts provides only a starting point to the development of a system-level model. The bottom up placement of these constituents into an environment that facilitates dynamic interaction representative of the real system completes the holistic view required to recognize system-level behaviors.

In a similar manner, cognitive architectures provide a holistic view of human information processing and decision making. The link between cognition, or micro specifications, and macro-level social phenomena can be more fully explored through cognitive social simulations. Starting with relatively simple cognitive models, of the type typically

proposed by Epstein, researchers can iteratively add complexity to the cognitive representation (Sun, 2006). Cognitive architectures consist of a system of systems replicating the human information processing system. The cognitive architecture itself takes a holistic view of human cognition. The use of group cognitive architectures based on social psychology could enable this same notion of multi-level modeling with cognitive social simulations.

C. A SAMPLE OF DEPARTMENT OF DEFENSE SIMULATION MODELS

This section provides a brief overview of DoD models and simulation identified as candidates for the application of reinforcement learning and or the inclusion of some form of cognitive architecture. We provide an introduction to the Combined Arms Analysis Toolkit for the Twenty-first Century (COMBATXXI), the Infantry Warrior Simulation (IWARS), the Assignment Scheduling Capability for Unmanned Aerial Systems (ASC-U), and the Cultural Geography model (CG).

1. Combined Arms Analysis Toolkit for the Twenty-first Century

COMBATXXI is a Joint high-resolution, closed-form, stochastic, analytical combat simulation developed at TRADOC Analysis Center -White Sands Missile Range (TRAC-WSMR) and the Marine Corps Combat Development Command, Operations Analysis Division (MCCDC-OAD) (Kunde, 2005). COMBATXXI is designed to support analysis at the brigade and below and supports the representation of light and heavy forces, air mobile forces, future force capabilities, fixed and rotary wing aircraft, and amphibious and urban operations. COMBATXXI has been used in support of a number of studies in support of acquisitions activities including analysis in support of the Ground Soldier Systems analysis of alternatives (AoA) and the current Ground Combat Vehicle AoA. COMBATXXI behaviors are implemented in the python programming language allowing great flexibility in the creation of new behaviors to drive entity actions. Recently, the use of hierarchical task networks have been used to provide COMBATXXI entities a GOMS like ability to achieve goals. A need exists to improve the representation of environmental sensing and

knowledge representation at the individual soldier-level within COMBATXXI and to have more adaptable agents in general.

2. UrbanSim

game-based training simulations provide practice environments that allow trainees to develop knowledge, skills, and abilities required for the execution of real-world tasks in a relatively risk-free and cost effective setting making them especially attractive for DoD applications. Challenges exist in the measurement of the effectiveness of these systems in transferring skill proficiency to real-world tasks and the validation and verification of the simulation model and its supporting data, though the requirements for verification and validation for a training use case are less stringent that those imposed on simulation models designed for analysis. Unique to training systems is a need to verify that the reinforcement provided by the training system supports the learning objectives of the system. As the systems become more complex understanding how the system will respond to all potential actions selected by a trainee can present challenges due to the size of the potential state space.

> The Army requires the capability to develop adaptive digitized learning products that employ artificial intelligence and/or digital tutors to tailor learning to the individual Soldiers experience and knowledge-level and provide a relevant and rigorous, yet consistent, learning outcome. (U.S. Army 2011)

UrbanSim is a DoD sponsored game-based training simulation designed to develop knowledge, skills, and abilities associated with tactical-level decision-making in counterinsurgency and irregular warfare operations developed at the Institute for Creative Technology (ICT). UrbanSim relies on an underlying multi-agent simulation, PsychSim also developed at ICT, to adjudicate the effect of actions taken by the trainee at each turn and to provide feedback to the player in the form of a numeric reward signal, graphically displayed as a horizontal bar chart showing the percent of the maximum score the player is currently at for each of the six categories that make up the aggregate score, ($\max = 600$). The standard training scenario packaged with the product is intended to train students to adopt a

clear, hold, build strategy by selecting the actions at each turn for eleven blue units at each of fifteen game turns. The game developers identified that the verification of all potential paths through the training system required the use of automated mechanisms (Wansbury, Hart, Gordon, & Wilkinson, 2010; Wang, Pynadath, & Marsella, 2012).

3. Assignment Scheduling Capability for Unmanned Aerial Systems

Unmanned systems form an increasingly important component of the surveillance and reconnaissance capability of the U.S. and have shown their value in the last decade of conflict in a variety of roles (Ahner, Buss, & Ruck, 2006). The efficient allocation of assets in theater remains challenging as does the analysis supporting future procurement of this class of systems. TRADOC Analysis Center (TRAC) conducted a UAS Mix study in 2006 and developed an assignment scheduling capability for unmanned aerial vehicles (ASC-U) at TRAC-Monterey, in partnership with the MOVES Institute and Rolands and Associates Inc. (Nannini, 2006).

ASC-U employs a discrete event simulation model coupled with the optimization of a linear objective function over a finite-time horizon. ASC-U determines a feasible schedule for UAV missions that can be successfully executed in a scenario with a specific mix of UAVs by obtaining an optimal solution to a simplified problem that assigns available UAVs to missions that are available or will be available within a future time horizon at set intervals. The need exists to improve the ability of the scheduler agent to develop near optimal plans schedules in cases with delayed and noisy rewards.

4. Cultural Geography Model

The Cultural Geogrphy (CG) model is a government owned, open source, agent based social simulation designed to represent the population in a brigade area of operations in an Irregular Warfare environment, see Figure 10 (J. Alt et al., 2009). CG model agents operate within a social network updating their stances on issues of interest as new information is received through observation or communications. The model is built on a conceptual model developed through interaction with experts from the social sciences and is intended

to be modular in nature with an architecture supporting the integration of behavioral and social modules as needed to support the modeling and analysis of a particular area of the world during a particular time period (J. Alt et al., 2009). The basic components of the CG model are the cognitive module, the social network module, and the infrastructure module. The cognitive module uses Bayesian networks to represent agent internal state, with agents representing population segments in most use cases.

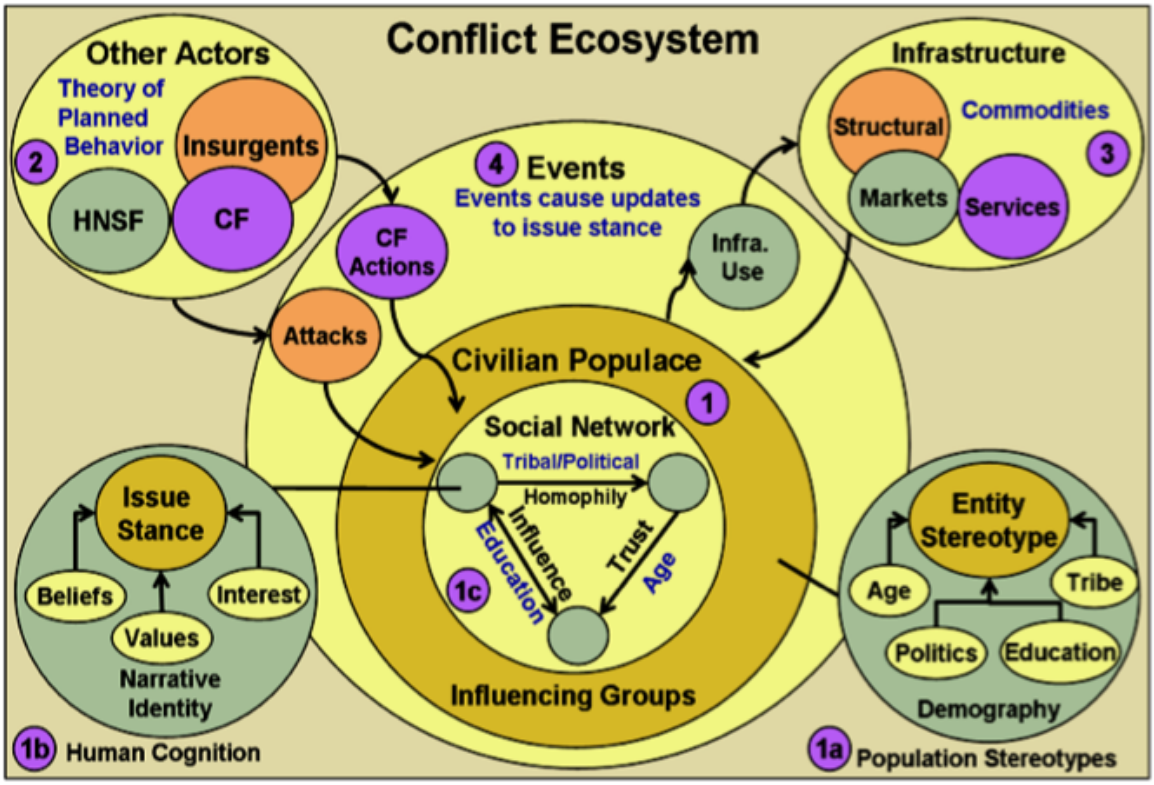


Figure 10: Cultural geography conceptual model.

The CG model does not specify a cognitive architecture. The social network module calculates the social distance between agents based on an implementation of the theory of homophily. The infrastructure module represents essential services as multi-server queues with finite capacity. Scenarios within the CG model are built around issues in the area under study that are relevant to the population and the stabilizing force. Data develop-

ment follows a process that mirrors the counter-insurgency intelligence preparation of the battlefield (Mansoor, 2007).

D. SUMMARY

In this chapter we provided relevant background material and reviewed the state of the art in reinforcement learning, cognitive architectures, and several currently used DoD simulation models. In the next chapter, we will present the novel use of an exponentially weighted average that makes use of continuous-time as an action-value estimator in a reinforcement learning system and document the it performance in several benchmark and game-based environments. In Chapter IV, we will provide an account of the application of this approach to three different DoD simulation use cases:

- ASC-U
- COMBATXXI
- UrbanSIM

In chapter five we will discuss the use of reinforcement learning and document a practical cognitive architecture, both applied to emerging DoD efforts in the area of human, social, cultural and behavioral modeling within the Cultural Geography model.

III. DIRECT-Q COMPUTATION

This chapter details a the novel use of an exponentially weighted average that makes use of continuous environment time as an action value estimator within a reinforcement learning agent and improves learning speed compared to dominant temporal differencing algorithms designed to address this problem. We develop the algorithm and provide results of empirical benchmarking in standard benchmark and game-based domains. Emerging results of the use of an early version of this algorithm in a variety of applications were previously reported in several conference proceedings, an accepted in-press journal article (S. Papadopoulos, Alt, Darken, & Baez, 2013), and student thesis work supported by this effort (J. K. Alt et al., 2011; Ozcan, Alt, & Darken, 2011; Ozkan, 2011; M. Papadopoulos, 2010; Pollock, Alt, & Darken, 2011).

A. DIRECT-Q COMPUTATION

Delayed rewards are common in RL making the ability to efficiently assign credit to (s, a) essential to developing an accurate estimate of $Q(s, a)$. The dominant approaches to address noisy and delayed rewards in RL make use of noisy hill-climbing approaches (temporal differencing). In this research, we attempt to determine if a relatively straightforward and direct approach that uses an intuitive estimator and better accounts for environmental change can result in improved performance over the dominant hill-climbing approaches to this problem. Our approach develops two key ideas:

- Instead of noisy hill-climbing can we make use of the obvious intuitive estimator, the historical reward?
- Instead of using the agent's action count, can we make use of the environment time?

DQ-C assigns credit for all future point rewards to each visit to a (s, a), developing an estimate based on the historical average of the long-term value of each attempt of a state-action pair. This long-term estimate uniquely makes use of environment time rather than

action counts in the development of this discounted term. These estimates are then used to calculate either the sample average or exponentially weighted average value of each (s, a). DQ-C's unique treatment of time allows it to adapt to changing environments more readily than similar algorithms that do not account for time, focusing instead on system updates to serve as a counter. In this section we develop both versions of the proposed approach. We define the following indices,

- i: rewards, $i = \{1..k\}$, where k is total number of rewards received,
- j: attempts of a (s, a), $j = \{1..n\}$, where n is the total number of attempts of a (s, a),

and the following additional terms,

- r_i: value of the i^{th} point reward,
- t_i: arrival time of the i^{th} point reward,
- τ_j: selection time of the j^{th} attempt of a (s, a),
- γ: discount factor, $\gamma \in (0, 1]$,
- α: learning rate, $\alpha \in [0, 1]$,
- t: current simulation time, $t = \{1..T\}$, where T is the maximum time in the simulation,
- $H(t_i - \tau_j)$: an indicator function that is equal to 0 if $t_i - \tau_j < 0$, 1 otherwise,

which we will use throughout this section. We define the expected utility of each j^{th} attempt of a (s, a) as the discounted sum of the point rewards, r_i, received following visit to the (s, a).

$$U_j(s, a) \equiv \sum_{i=1}^{k} r_i \gamma^{t_i - \tau_j} H(t_i - \tau_j), \tag{25}$$

where $H(t_i - \tau_j)$ is an indicator function that ensures that only point rewards received following the attempt of a (s, a) are credited to the j^{th} attempt, equal to 0, if $t_i - \tau_j < 0$, or 1, if $t_i - \tau_j > 0$. Each future point reward is discounted back to the time at which the (s, a) was visited. So we see that as $(t_i - t_j) \to \infty$ that the contribution of additional point rewards to this term goes to zero. Note that this does not depend on the number of k point rewards, only the time difference. Next we estimate the long-term value of each (s, a) by calculating the expected utility of each (s, a) using either a sample average,

$$Q(s, a) = \frac{1}{n} \sum_{j=1}^{n} U_j(s, a), \tag{26}$$

or an exponentially weighted average,

$$Q(s, a) = \frac{\sum_{j=1}^{n} \alpha^{t-\tau_j} U_j(s, a)}{\sum_{j=1}^{n} \alpha^{t-\tau_j}} = \frac{\sum_{j=1}^{n} \alpha^{-\tau_j} U_j(s, a)}{\sum_{j=1}^{n} \alpha^{-\tau_j}}, \tag{27}$$

Both versions can be updated upon arrival of point rewards or in a continuous manner. We will first discuss the recursive update to $U_j(s, a)$ upon the arrival of the $k+1$ point reward. First define the utility of the j^{th} attempt of a state-action pair, (s,a), as a function of the k point rewards received, $U_j(k) \equiv U_j(s, a)$. We can see that the expected utility of each j^{th} attempt of a state-action pair, $U_j(s, a)$, only changes with the arrival of point rewards, so in order to update our estimate of the value of each attempt we need to update $U_j(k + 1)$,

$$U_j(k + 1) = U_j(k) + r_{k+1} \gamma^{t_{k+1} - \tau_j} H(t_{k+1} - \tau_j). \tag{28}$$

In order to update the overall estimate of the value of a state-action pair, $Q(s, a)$, we first define it as a function of the number of attempts of each state-action pair, n, and the number of point rewards, k, received, $Q(n, k) \equiv Q(s, a)$.

We can think of the update needing to occur in two cases. In the first case, we have received no additional point rewards, but have increased our number of attempts of a state-action pair, $Q(n+x,k)$,

$$Q(n+x,k) = \frac{n}{n+x}Q(n,k). \tag{29}$$

In the second case, we have received additional point rewards and have made additional attempts of a state-action pair, $Q(n+x,k+1)$,

$$Q(n+x,k+1) = \frac{n}{n+x}Q(n,k) + \frac{1}{n+x}\sum_{j=1}^{n+x} r_{k+1}\gamma^{t_{k+1}-\tau_j}H(t_{k+1}-\tau_j). \tag{30}$$

Note that the first half of this term is identical to the update for $Q(n+x,k)$ and that in the second half of this term we simply update the value of all attempts, $j = 1...n+x$, of a state-action pair with the $k+1$ reward, making sure that we weight this incremental update equally with the previous by multiplying by $\frac{1}{n+x}$. Since all previous attempts receive credit for new rewards, we must update all j visits rather than just the $n...n+x$ attempts. This produces a straightforward and direct estimate that is equally weighted and takes into account environment time in the update of the expected utility of each visit, but does not adjust the weights on each visit based on environment time, see Algorithm 5. We make a decision in the Algorithm 5 to conduct updates only on the receipt of point rewards, however, the algorithm could be easily adjusted to make updates following each attempt of a state-action pair with no receipt of a point reward. The choice depends on the needs of the application and computational considerations. We can make this estimate more sensitive to changes in the environment by developing the second key idea further by incorporating an exponentially weighted average.

Algorithm 5 Direct-Q Computation (DQ-C) with sample averaging

Parameters: discount rate, γ; base weight, α.
Indices: i, index of point rewards, $i \in \{1 \ldots k\}$; j, index of attempts of a state-action pair, $j \in \{1 \ldots n\}$
Terms: number of attempts of a state-action pair, n; number of attempts of a state-action pair since last update, x; number of point rewards received, k; time of attempt of j^{th} state-action pair, τ_j; value of the i^{th} point reward, r.
Initialize π (ie. ϵ-greedy, Boltzmann).
Initialize for all state-action pairs as they are encountered: Q(s,a)=0, estimate of each state-action pair; n(s,a)=0, counter for number of attempts of a state-action pair; x(s,a)=0, counter for number of attempts of a state-action pair since last update.
while agent is running **do**
  $t \leftarrow$ current time
  Return a using $\pi(s)$.
  $\tau_j \leftarrow t$
  Append τ_j to list of most attempt times for (s, a).
  $x(s, a) \leftarrow x(s, a) + 1$
  Observe for point reward, r.
  if point reward r is observed **then**
    for all state-action pairs observed, update Q(s,a) **do**
      if n(s,a) not 0: **then**
        $x \leftarrow x(s, a)$
        $n \leftarrow n(s, a)$
        $Q(s, a) \leftarrow \frac{n}{n+x} Q(s, a)$
        $n \leftarrow n + x$
        $n(s, a) \leftarrow n$
        $x(s, a) \leftarrow 0$
        for j from 1 to n **do**
          $Q(s, a) \leftarrow Q(s, a) + \frac{1}{n} r\gamma^{t-\tau_j}$
        end for
      end if
    end for
  end if
end while

Considering the update of the exponentially weighted version we see that there is no change to the update of the expected utility, $U_j(k)$, of each j^{th} visit to a state-action pair from the previous update. In order to update the overall estimate of the value of a state-action pair, we first define it as a function of the number of attempts of each state-action pair, n, the number of point rewards, k, and the current time, t, $Q(n, k, t) \equiv Q(s, a)$. We choose to leave the current time in this version of the formulation since it allows us to avoid potential numerical precision errors and we provide an alternative without it subsequently.

In this case, since the weights applied to each j^{th} visit are dependent on the current time, t, we must update the value of each set of weights upon each update, see Algorithm 6. Here we conduct an update to the numerator for each state-action pair as a function of the number of attempts of a state-action pair, the number of point rewards, and the current time, $N(n,k,t) \equiv N(s,a)$, and the denominator for each state-action pair as a function of the number of attempts of each state-action pair and the current time, $D(n,t) \equiv D(s,a)$. Let,

$$\begin{aligned} N(n,k,t) &\equiv \sum_{j=1}^{n} \alpha^{t-\tau_j} U_j(k), \\ D(n,t) &\equiv \sum_{j=1}^{n} \alpha^{t-\tau_j}, \\ Q(n,k,t) &\equiv \frac{N(n,k,t)}{D(n,t)}. \end{aligned} \tag{31}$$

Rearranging terms, we isolate our time dependent parameter in both the numerator and denominator,

$$\begin{aligned} N(n,k,t) &= \alpha^t \sum_{j=1}^{n} \frac{1}{\alpha^{\tau_j}} U_j(k), \\ N(n,k,t) &= \alpha^t S(n,k), \end{aligned} \tag{32}$$

where,

$$S(n,k) \equiv \sum_{j=1}^{n} \frac{1}{\alpha^{\tau_j}} U_j(k), \tag{33}$$

and,

$$D(n,t) = \alpha^t \sum_{j=1}^{n} \frac{1}{\alpha^{\tau_j}}$$
$$D(n,t) = \alpha^t C(n), \tag{34}$$

where,

$$C(n) \equiv \sum_{j=1}^{n} \frac{1}{\alpha^{\tau_j}}. \tag{35}$$

Now we can state the update for $N(n,k,t)$ as,

$$N(n+x,k+1,t) = \alpha^t S(n+x,k+1),$$
$$S(n+x,k+1) = S(n,k) + \sum_{j=1}^{n+x} \frac{1}{\alpha^{\tau_j}} r_{k+1} \gamma^{t_{k+1}-\tau_j} H(t_{k+1}-\tau_j), \tag{36}$$

and the update for $D(n,t)$ as,

$$D(n+x,t) = \alpha^t C(n+x),$$
$$C(n+x) = C(n) + \sum_{j=n+1}^{n+x} \frac{1}{\alpha^{\tau_j}}. \tag{37}$$

DQ-C uses all available data and by employing exponential weighting it allows the use of DQ-C in an on-line manner. The exponential weighting also allows DQ-C to be sensitive to changes in the environment, while avoiding the use of TD methods with their inherent assumption regarding the Markov property, which seldom holds in application environments. In Algorithm 6 we make use of the above formulation. See Appendix D for a brief theoretical analysis.

Algorithm 6 Direct-Q Computation (DQ-C) with exponential weighting

Parameters: discount rate, γ; base weight, α.
Indices: i, index of point rewards, $i \in \{1 \ldots k\}$; j, index of attempts of a state-action pair, $j \in \{1 \ldots n(s,a)\}$
Terms: number of attempts of a state-action pair, n; number of point rewards received, k; arrival time of i^{th} point reward, t_i; time of attempt of j^{th} state-action pair, τ_j; value of the i^{th} point reward, r_i.
Initialize π (ie. ϵ-greedy, Boltzmann).
Initialize for all state-action pairs as they are encountered: Q(s,a)=0, estimate of each state-action pair; N(s,a)=0, numerator used in estimate, and the current time; D(s,a)=0, the denominator of the estimate.
while agent is running **do**
  $t \leftarrow$ current time
  Return a using $\pi(s)$.
  $\tau_j \leftarrow t$
  Append τ_j to list of attempt times for (s,a).
  $n(s,a) \leftarrow n(s,a) + 1$.
  Observe for point reward, r.
  if point reward is observed **then**
    for all state-action pairs observed, update Q(s,a) **do**
      if n(s,a) not 0: **then**
        $n \leftarrow n(s,a)$
        for j from 1 to n **do**
          $N(s,a) \leftarrow N(s,a) + \frac{1}{\alpha^{\tau_j}} r \gamma^{t-\tau_j}$
          $D(s,a) \leftarrow D(s,a) + \frac{1}{\alpha^{\tau_j}}$
        end for
        $Q(s,a) \leftarrow \frac{N(s,a)}{D(s,a)}$
      end if
    end for
  end if
end while

In the next section we will discuss the results of empirical benchmarking with DQ-C.

B. BENCHMARK PROBLEMS

We compare DQ-C to Q-learning, SARSA, SARSA(λ), and Q(λ) using ϵ-greedy and Boltzmann policies on the n-arm bandit task as a benchmark for learning speed in a simple single state environment. We next compare the performance of DQ-C to Q-learning, SARSA, SARSA(λ), and Q(λ) in gridworlds ranging in size from 2x2 to 10x10 using ϵ-greedy and Boltzmann policies across a range of input parameters and environmental

conditions in each case. In each case parameters were tuned for each algorithm to ensure optimal performance and the results reported reflect the best performance of each in the given task environment.

1. N-Arm Bandit

Bandit problems can be described as single state MDPs $\langle s_0, a_n \in A, R(s_0, a_n)\rangle$ (Robbins, 1952). The problem is to determine the optimal policy to maximize long-term utility, $\pi^*(s_0)$. Performance in bandit problems can be measured using regret, the cost of not choosing the best arm,

$$\rho = T\mu^* - \sum_{t=1}^{T} \mu_{i \in n}, \tag{38}$$

accumulated over a finite-time horizon, T, where μ is the expected return of an arm and μ^* is the expected return of the best arm (Kocsis & Szepesvári, 2006). We treat this as a continuous learning task for 250 trials, where each trial is an attempt or pull of an arm. At the end of each trial the total accumulated regret measured and the mean and standard error of the sample of 500 replications is reported. Per Sutton's description of the 10-arm bandit benchmark, we examine a case with zero noise on the reward, $\sigma^2 = 0$, and a case with gaussian noise, $\sigma^2 = 1$.

In the 10-arm benchmark task with zero noise on the reward, $\sigma^2 = 0$, SARSA paired with Boltzmann, β, demonstrated the lowest total regret, see Figure 11, followed closely by Q-learning, and DQ-C paired with ϵ-greedy, see Table 1. The results for the 10-arm case with noise on the reward, $\sigma^2 = 1$, are similar with DQ-C, paired with β leading folllowed by Q-learning and SARSA, both paired with ϵ-greedy, see Table 2. DQ-C performs well in the most challenging of the bandit environments explored, the 10-arm benchmark with noise, accumulating 23% less regret that Q-learning, its nearest competitor.

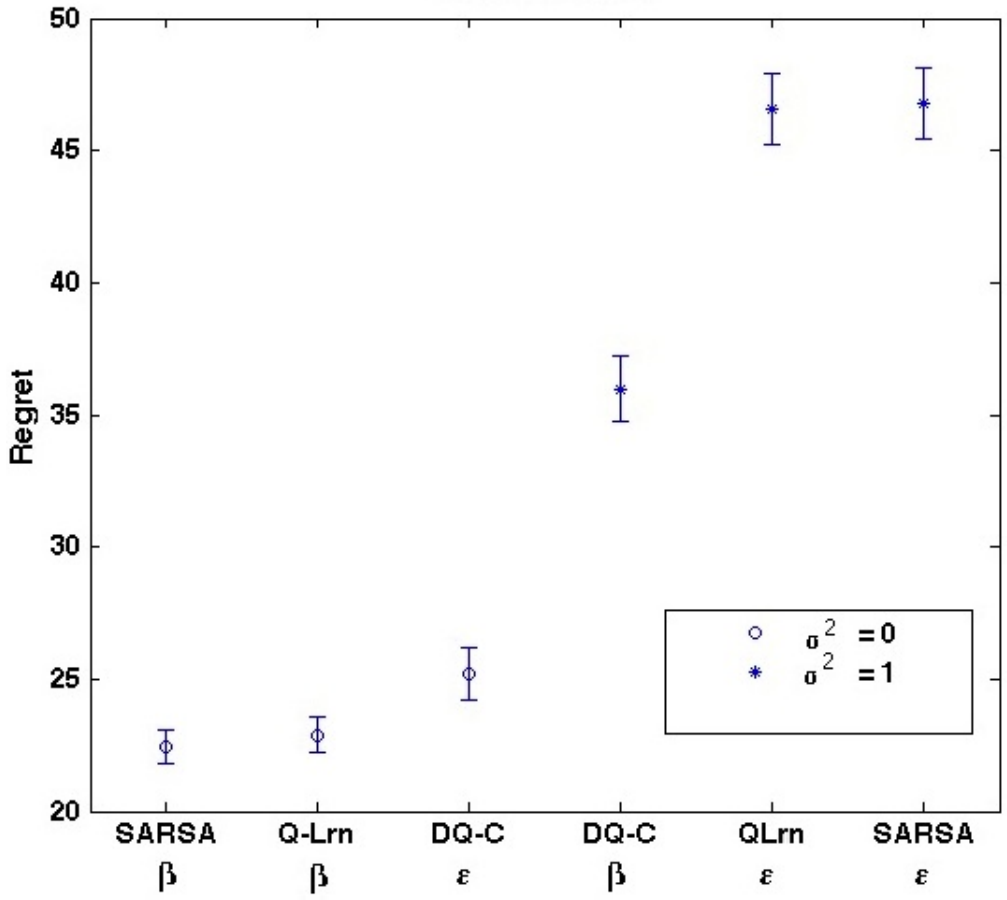


Figure 11: Top three performers for 10-arm bandit benchmark domain, for $\sigma^2 = 1.0$ and $\sigma^2 = 0.0$, 250 trials and 500 replications. Mean total regret is plotted for each algorithm policy pair along with associated standard error.

Table 1: Regret results for algorithms coupled with $epsilon - greedy$, β, for 10-arm bandit benchmark task following 250 trials with 500 replications with σ^2=0 on the reward signal. SARSA with ϵ-greedy provided best result.

ALGORITHM	ϵ	β
Q-LEARNING	69.52±1.23	22.89±0.66
SARSA	71.17± 1.15	**22.43±0.65**
Q(λ)	44.4± 1.46	87.08±1.59
SARSA(λ)	55.23± 1.86	77.38± 2.22
DQ-C	25.21± 0.97	55.85±1.0

Table 2: Regret results for algorithms coupled with $epsilon - greedy$, β, for 10-arm bandit benchmark task following 250 trials with 500 replications with σ^2=1 on the reward signal. DQ-C with β provided best results.

ALGORITHM	ϵ	β
Q-LEARNING	46.56±1.35	67.45±1.56
SARSA	46.77±1.35	63.94±1.22
Q(λ)	65.45±1.71	53.44±1.88
SARSA(λ)	60.8±1.7	53.07±1.8
DQ-C	47.01±1.34	**35.99±1.27**

2. Two-Arm Bandit

In order to further understand the performance of the algorithms in the most simple cases we also used the same procedures to examine the 2-arm bandit case. In this set of experiments we also explored the impact of the absolute difference between the mean of the two arms. When no noise is present DQ-C produced the best performance when the arms were well separated and was slightly outperformed in the case where the arms were only 0.2 apart by Q-learning and SARSA, see Tables 3. Since Q-learning systematically

overestimates the best arm, this strategy pays off when the true values of the arms are close (Thrun & Schwartz, 1993). In the case where there is noise on the arms Q(λ) produces lower regret than DQ-C in the case where the means of the arms are well separated, but in the most challenging case, low separation on the arms with noise, DQ-C produces the best result, see Table 4.

Table 3: Regret results for algorithms coupled with $epsilon - greedy$ and β for 2-arm bandit following 250 trials with 500 replications with $\sigma^2 = 0$, $\delta = \mu^* - \mu_1 = 0.8$ and 0.2. DQ-C with β provides best results in $\delta = 0.8$ case, Q-learning and SARSA paired with $\epsilon - greedy$ in the $\delta = 0.2$ case.

ALGORITHM	$\epsilon, \delta = 0.8$	β
$aceQ-learning$	14.59±0.44	30.83±0.64
SARSA	14.59±0.44	19.81±0.97
Q(λ)	65.45±1.71	4.96±0.26
SARSA(λ)	73.68±1.44	6.36±0.48
DQ-C	17.73±0.57	**1.74**±0.08
ALGORITHM	$\epsilon, \delta = 0.2$	β
Q-LEARNING	**3.64**±0.11	21.66±0.20
SARSA	**3.64**±0.11	21.53±0.21
Q(λ)	18.87±0.40	5.74±0.27
SARSA(λ)	13.97±0.12	15.15±0.620
DQ-C	4.52±0.14	5.25±0.26

Table 4: Regret results for algorithms coupled with $epsilon - greedy$ and β for 2-arm bandit following 250 trials with 500 replications with $\sigma^2 = 1$, $\delta = \mu^* - \mu_1 = 0.8$ and 0.2. $Q(\lambda)$ paired with β resulted in the best performance for $\delta = 0.8$ and DQ-C paired with β for $\delta = 0.2$.

ALGORITHM	$\epsilon, \delta = 0.8$	β
Q-LEARNING	14.49±0.52	20.95±0.24
SARSA	14.53±0.52	19.1±0.32
$Q(\lambda)$	19.05±0.58	**9.28**±0.41
SARSA(λ)	18.48±0.75	14.79±0.80
DQ-C	14.28±0.52	12.33±0.65
ALGORITHM	$\epsilon, \delta = 0.2$	β
Q-LEARNING	14.81±0.41	31.96±0.75
SARSA	14.79±0.40	19.53±0.85
$Q(\lambda)$	35.47±1.38	15.13±1.18
SARSA(λ)	21.71±1.13	11.57±0.89
DQ-C	14.68±0.42	**4.62**±0.37

3. Gridworld

Gridworld provides a sequential MDP, a stochastic shortest path problem that can be described by the tuple $\langle s \in S, a_n \in A, P^a_{s,s'}, R(s_0, a_n)\rangle$, see Figure 12.

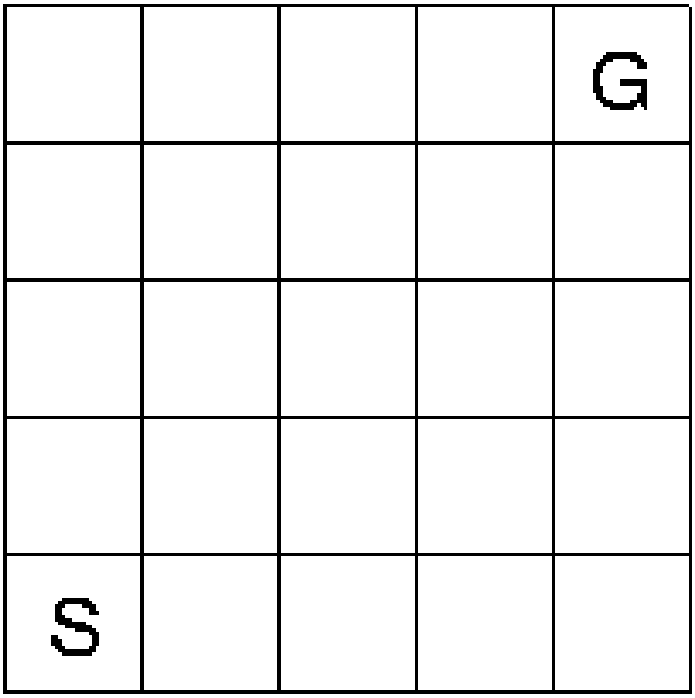


Figure 12: Sample gridworld domain.

We initially explore performance in deterministic and stochastic gridworlds varying in size from 2x2 to 10x10. In the stochastic version of this task, we use a simple model-based on the grid world from Russell and Norvig, but maintain only a single positive reward upon attainment of the goal state (Russell & Norvig, 2010). The transition matrix for each state is $P_a^{s,s'} = 0.8$ and $P_a^{s,s} = 0.2$.We measure performance by the mean number of times the goal state was attained over 1000 replications of each condition. In each case the trial period was limited to 500 time steps, meaning the agent sought to maximize the number of times the goal was attained during that period. In both the deterministic case and the stochastic case, DQ-C paired with either ϵ-greedy or Boltzmann provided a greater overall utility, over two times the nearest competitor, than the comparison cases for the same time periods indicating that DQ-C learned faster in this benchmark domain, see Figures 13 and 14 as well as Tables 5 and 6.

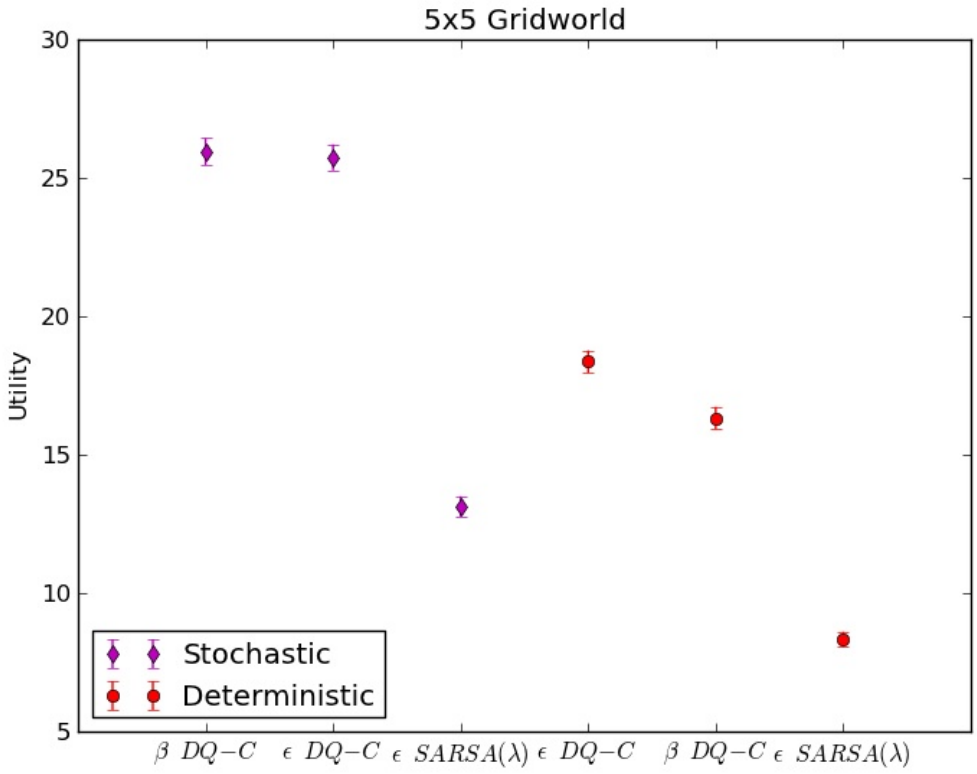


Figure 13: Top three performers for deterministic and stochastic 5x5 gridworld task, 500 trials and 1000 replications. Mean total utility for each is plotted for each algorithm policy pair along with associated standard error.

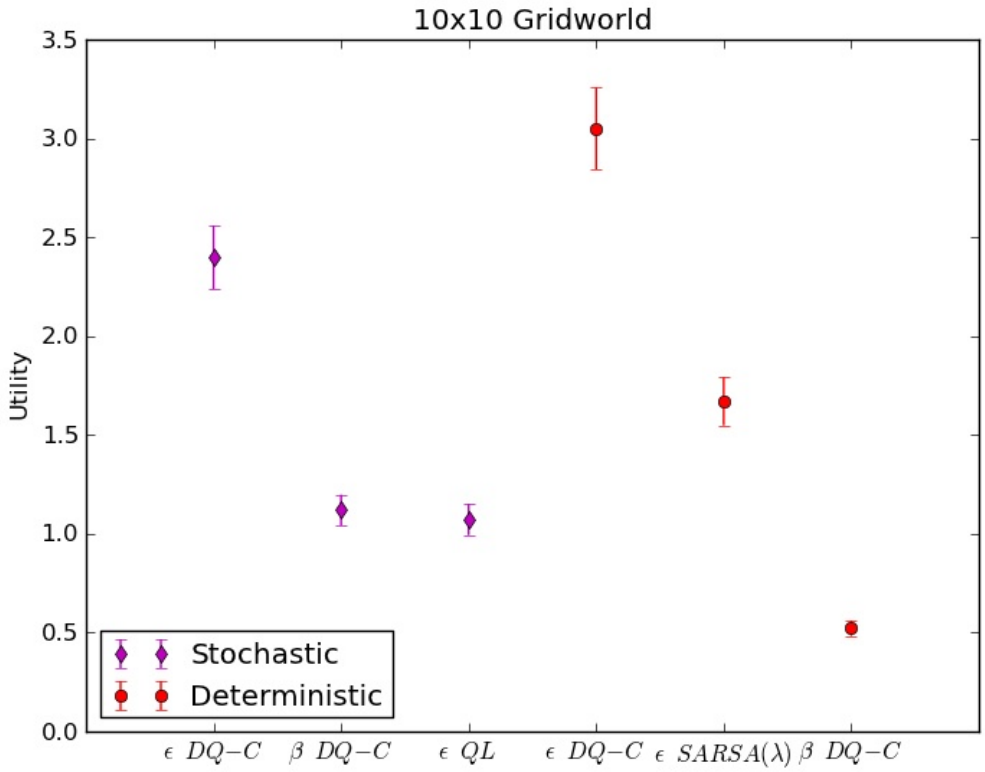


Figure 14: Top three performers for deterministic and stochastic 10x10 gridworld task, 500 trials and 1000 replications. Mean total utility for each is plotted for each algorithm policy pair along with associated standard error.

Table 5: Mean goals achieved for deterministic (top) and stochastic (bottom) 5x5 grid world following 500 trials with 1000 replications with single reward upon attainment of the goal state by algorithm and policy. DQ-C with ϵ-greedy obtained the best result in each case.

ALGORITHM	ϵ	β
Q-LEARNING	3.84± 0.12	3.04± 0.08
SARSA	3.734± 0.12	2.93±0.07
Q(λ)	2.23± 0.05	2.17± 0.55
SARSA(λ)	7.34± 0.22	5.97± 0.16
DQ-C	**15.25± 0.33**	14.45±0.33
Q-LEARNING	2.508± 0.08	2.24± 0.06
SARSA	2.49± 0.08	2.24±0.06
Q(λ)	1.74± 0.05	1.77± 0.05
SARSA(λ)	4.53± 0.14	4.28±0.14
DQ-C	**10.30± 0.28**	9.34±0.27

Table 6: Mean goals achieved for deterministic (top) and stochastic (bottom) 10x10 grid world following 500 trials with 500 replications with single reward upon attainment of the goal state by algorithm and policy. DQ-C with ϵ-greedy obtained the best result in each case.

ALGORITHM	ϵ	β
Q-LEARNING	0.504±0.03	0.586±0.02
SARSA	0.504± 0.02	0.576±0.02
Q(λ)	0.484± 0.02	0.544± 0.02
SARSA(λ)	1.67± 0.12	0.668±0.03
DQ-C	**3.03± 0.21**	1.12±0.07
Q-LEARNING	0.47± 0.02	0.39± 0.02
SARSA	0.47± 0.02	0.394±0.02
Q(λ)	0.45± 0.02	0.384± 0.02
SARSA(λ)	1.07± 0.07	0.432±0.02
DQ-C	**2.39± 0.16**	0.524±0.03

In order to gain further insight into the performance of DQ-C under more challenging conditions we repeated the procedures described above on a non-stationary stochastic version of gridworld, where the likelihood of transitioning from a state oscillated according to a sinusoidal function and a version of grid world with a dynamic goal. In both cases DQ-C outperformed alternative methods, with DQ-C paired with both exploration techniques finishing in the top two slots in each case followed by SARSA(λ), see Figure 15.

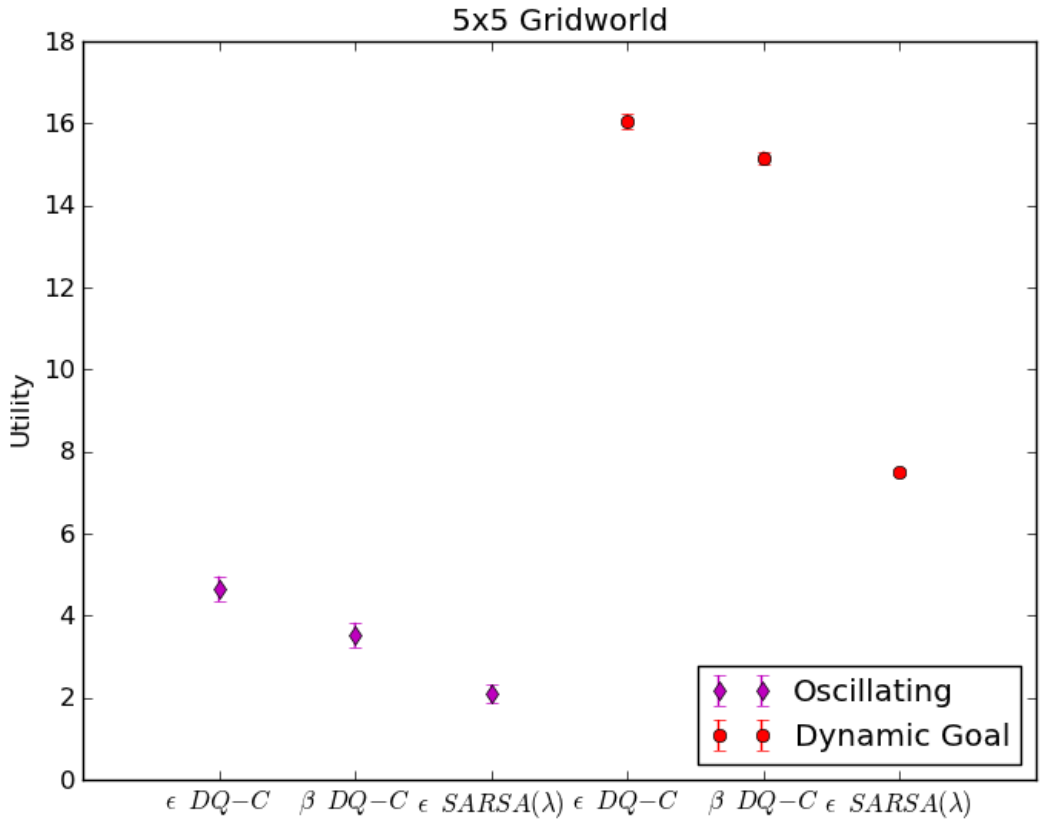


Figure 15: Top three performers for 5x5 gridworld, with oscillating transition matrix and a dynamic goal, 250 trials and 500 replications. Mean utility is plotted for each algorithm policy pair along with associated standard error.

The results from the dynamic gridworld illustrate the success of the algorithm in performing well in non-stationary environments.

4. Summary

Our estimator results in faster learning on the most challenging empirical cases that we have explored, those with noisy and delayed rewards, with noticeable improvements relative to dominant algorithms under two of the more challenging conditions. Future work

will compare the performance of DQ-C to model-based methods, conduct further analysis of its theoretical properties, and document its application in applied domains. See Appendix I for additional benchmarking results.

In the next sections, we will examine how the results we observed in these benchmark domains transition over to more challenging applied problems in game domains. The first section details the use of RL in a game-based domain, the Physical Traveling Salesman Problem. The second section provides results of the incorporation of an RL controller into the classic arcade game, Pacman.

C. PHYSICAL TRAVELING SALESMAN PROBLEM

The traveling salesman problem (TSP) is one of the most widely studied optimization problems with many approaches involving heuristics and meta-heuristics for finding approximately optimal solutions for for problems of very large scale involving many thousands of cities. The objective of a standard TSP is to minimize the total distance travelled. The physical traveling salesman problem (PTSP) makes a modification to the base problem that makes the solution to even relatively small problems much more complex. In this modified version the salesman has mass and moves by applying a force vector to the mass at each point in time and controlling the orientation of the mass through direction controls. The objective is to find the shortest path through the cities, measured by the total time required to complete the circuit, while minimizing the number of force vectors applied. The environment is a two-dimensional board with ten waypoints and multiple obstacles, see Figure 16. This problem is currently part of an open competition involving two IEEE sponsored conferences: the 2012 IEEE World Congress on Computational Intelligence and the 2012 IEEE Conference on Computational Intelligence and Games (http://www.ptsp-game.net).

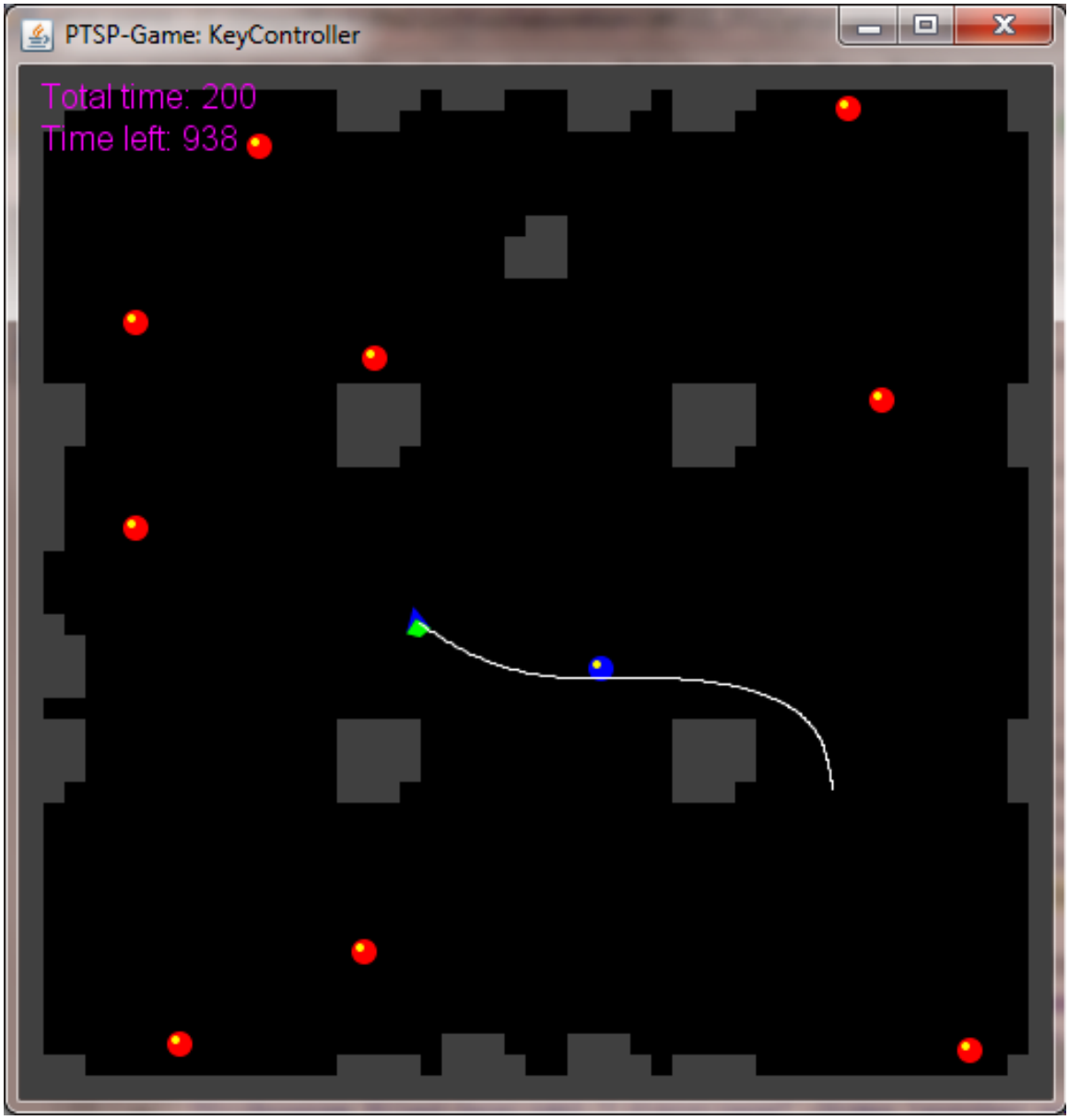


Figure 16: Sample map for physical traveling salesman problem.

1. Problem Specification

The PTSP requires that an object with mass, referred to as a ship, moving in a two dimensional environment with physics based laws of motion travel through a set of

waypoints while minimizing time with a lesser weight on the minimization of thrust. The controller must be initialized within 1000ms and at each iteration must respond with a control input within 40ms. We can describe the environment by defining the following indices,

- t, time, $t = \{1...T\}$, where T is the maximum time allowed to reach a waypoint, constant at T=1000 steps
- i, waypoint, $i = \{1..|W|\}$, where W is the set of waypoints that the ship must visit,
- j, action index, $j = \{0, 1, 2, 3, 4, 5\}$,

and the following additional terms,

- p_t, the ship's position vector in two dimensional space at time t,
- o_t, the ship's orientation at time t,
- v_t, the ship's velocity at time t,
- α, the rotation step-size applied to the ships orientation upon a control action, constant at $\alpha = \frac{\pi}{60}$ radians,
- L, the friction factor applied to the ships velocity at each t, constant at $L = 0.99$,
- K, the collision factor modifying the direction of the ship upon collision with an object, constant at $K = 0.25$
- $T_{initialize}$, the maximum time allowed for a controller to initialize, constant at $T_{initialize} = 1000$ms
- $T_{response}$, the maximum time allowed for a controller to respond at each t, constant at $T_{response} = 40$ms,

which are used in the following update equations,

$$\begin{aligned} p_{t+1} &= p_t + v_{t+1}, \\ o_{t+1} &= \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix}, \\ v_{t+1} &= (v_t + (o_{t+1} T_t K))L. \end{aligned} \tag{39}$$

The set of actions, $a \in A$, controls steering and the application of thrust to the ship and all actions act as forces applied to the ship to update its state using the update equations. Since our focus is RL we decompose the PTSP into a path planning problem, where the goal is to select the next waypoint in an efficient manner, and a ship control problem, where the goal is to learn how to use the six actions available to navigate the ship to its destination.

In order to determine which waypoint to navigate to next, we determine the closest waypoint from our current location that has not yet been visited, see Algorithm 7. If we have line of sight to the waypoint then the RL controller must control the ship to reach the waypoint, but if there is not line of sight to the next waypoint we locate the nearest location that does have line of sight to the waypoint and create an intermediate objective. In this case the RL controller first attempts to control the ship to reach the intermediate objective then re-assigns the destination to the target waypoint. Should the ship grossly overshoot its target, completely possible in this domain, it will identify the closest waypoint and reassign the waypoint objective. This is a purely greedy strategy, but serves its purpose in terms of providing a simple mechanism to select the next waypoint to facilitate the learning of the control task.

Algorithm 7 Greedy city selection

1: Create city list.
2: Set closest unvisited city as next waypoint.
3: **if** no line of site **then**
4: Set intermediate waypoint.
5: **end if**
6: Select control actions with RL controller.
7: Increment t. Go to line 2.

2. RL Formulation

We focus the RL formulation on the control subproblem of the PTSP. We define the RL control problem by the following tuple, $\langle \overrightarrow{s} \in S, a \in A, P_a^{s,s'}, R(s) \rangle$. The state description vector,s_t, $\overrightarrow{s_t} = \langle O_t, V_t, D_t \rangle$, describes the state of the ship at time t, where $O_t(o_t) \equiv$ orientation of ship relevant to goal, $V_t(v_t) \equiv$ speed, $D_t(d_t) \equiv$ distance from goal, and the state space is developed as described below.

$$O_t(o_t) = \begin{cases} 4 & o_t < -0.5 \\ 3 & -0.5 < o_t \leqslant 0 \\ 2 & 0 < o_t \leqslant 0.75 \\ 1 & 0.75 < o_t \leqslant 0.97 \\ 0 & 0.97 < o_t \end{cases}$$

$$V_t(v_t) = \begin{cases} 0 & 0 < v_t \leqslant 0.4 \\ 1 & 0.4 < v_t \end{cases}$$

$$D_t(d_t) = \begin{cases} 0 & 0 < d_t \leqslant 15 \\ 1 & 15 < d_t \end{cases}$$

The reward function is defined,

$$R(O_t, V_t, D_t) = \begin{cases} O_t \frac{1}{d_{t-1}+1} & 1 > O_t \& d_{t-1} - d_t > 0.4 \\ 0 & o.w, \end{cases}$$

and provides a graduated reward structure, but with a large bonus obtained only upon reaching the waypoint.

The challenge of developing the appropriate state space for a given problem is a significant research topic in its own right.

3. Empirical Results

In order to gain insight into the performance of the algorithm policy pairs on the control task we conducted computational experiments comparing the performance of DQ-C, SARSA, Q-Learning, SARSA(λ), and Q(λ) when paired with ϵ-greedy and β. We evaluated performance across a range of parameters for each algorithm and show results for the best performance observed for each algorithm. Each combination is replicated 1000 times, with each replication using a different initial random seed as a variance reduction measure. Additionally, performance is measured across ten maps of varying complexity, see Appendix E.

DQ-C matches or outperforms the comparison cases across all map sets. DQ-C paired with Boltzmann, β, had the highest number average number of waypoints found and the lowest time per waypoint, see Figure 17 and Figure 18. Q(λ) paired with Boltzmann followed DQ-C, and in the case of at least one map outperformed DQ-C. The RL task is formulated as a sequential task where the agent tries to learn how to control the ship to reach the next waypoint. As a result of this even during a single trip to a single city, there is an episodic nature to the task. Since the state is always described in terms of the ship's location relative to the next waypoint, the state-space is reduced significantly. DQ-C requires both γ and α to be specified and is designed to address the temporal credit assignment problem. Interestingly the other top performers in this game environment, Q(λ)

and SARSA(λ), both include the notion of eligibility traces, also intended to address the credit assignment problem (Sutton & Barto, 1998).

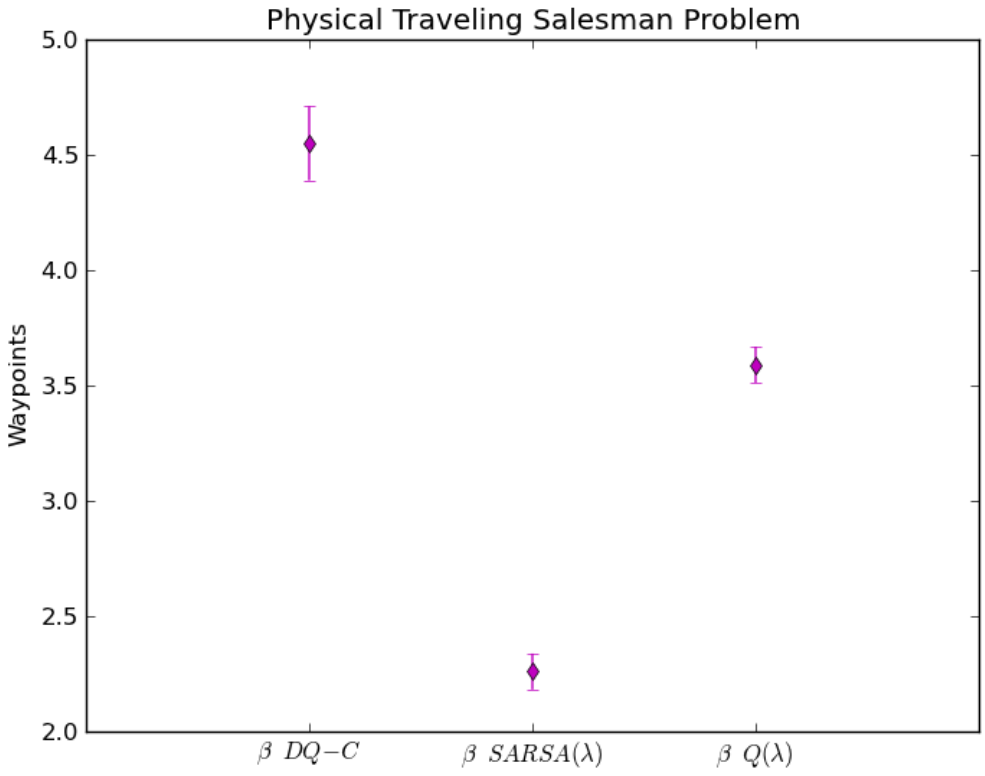


Figure 17: Mean waypoints obtained across all maps by top three performing algorithm and policy pairs.

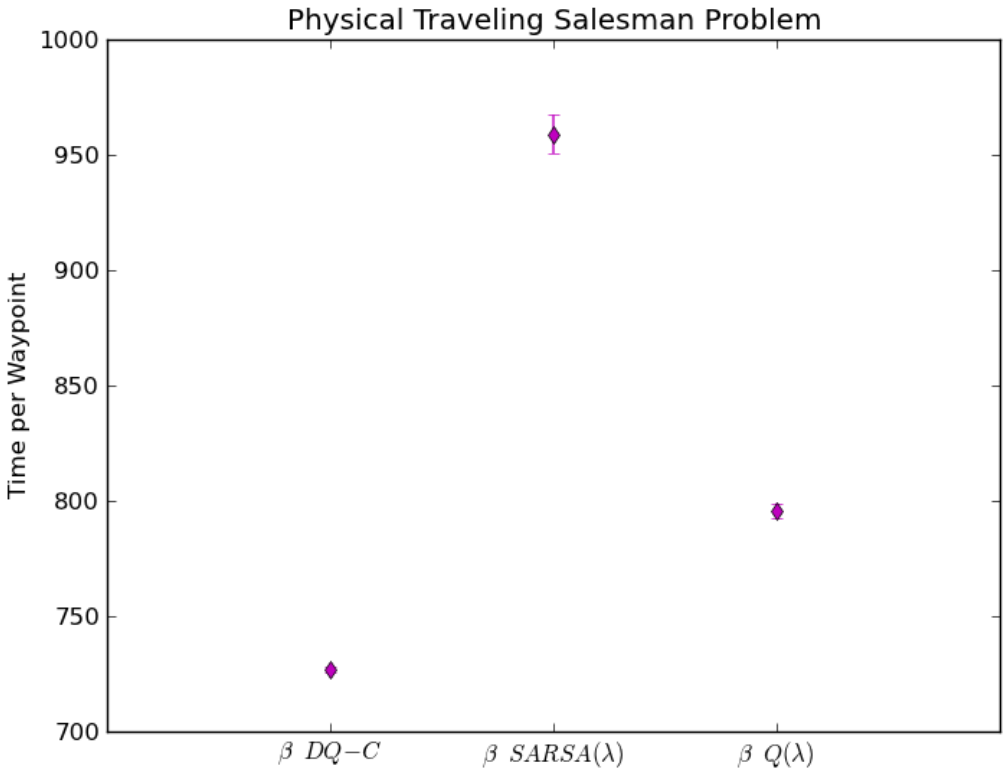


Figure 18: Mean time per waypoints obtained across all maps by top three performing algorithm and policy pairs.

4. Insights from the Physical Traveling Salesman Problem

As we saw in grid world and the 10-arm bandit, DQ-C continues to match or outperform the comparison cases across all map sets. DQ-C paired with Boltzmann had the highest number average number of waypoints found and the lowest time per waypoint. Q(λ) paired with Boltzmann followed DQ-C. The reward structure here was graduated to facilitate faster learning given the large state space, with the large reward attained upon reach each waypoint serving as a large delayed reward compared to the magnitude of the

interim rewards. This is an example of an environment that makes use of continuous-time rather than a turn based counter.

D. PACMAN

The Pacman arcade game places a player controlled agent, Pacman, in a variety of mazes with the goal of avoiding ghosts, two in this version, while consuming all dots on the board. The player has access to two power dots, which when eaten allow the ghosts to be temporarily consumed by Pacman. We build on the infrastructure provided by the Berkley Pacman projects with our empirical work focused on the medium sized classic map, see Figure 19, (http://www.inst.eecs.berkeley.edu/cs188/pacman/pacman.html).

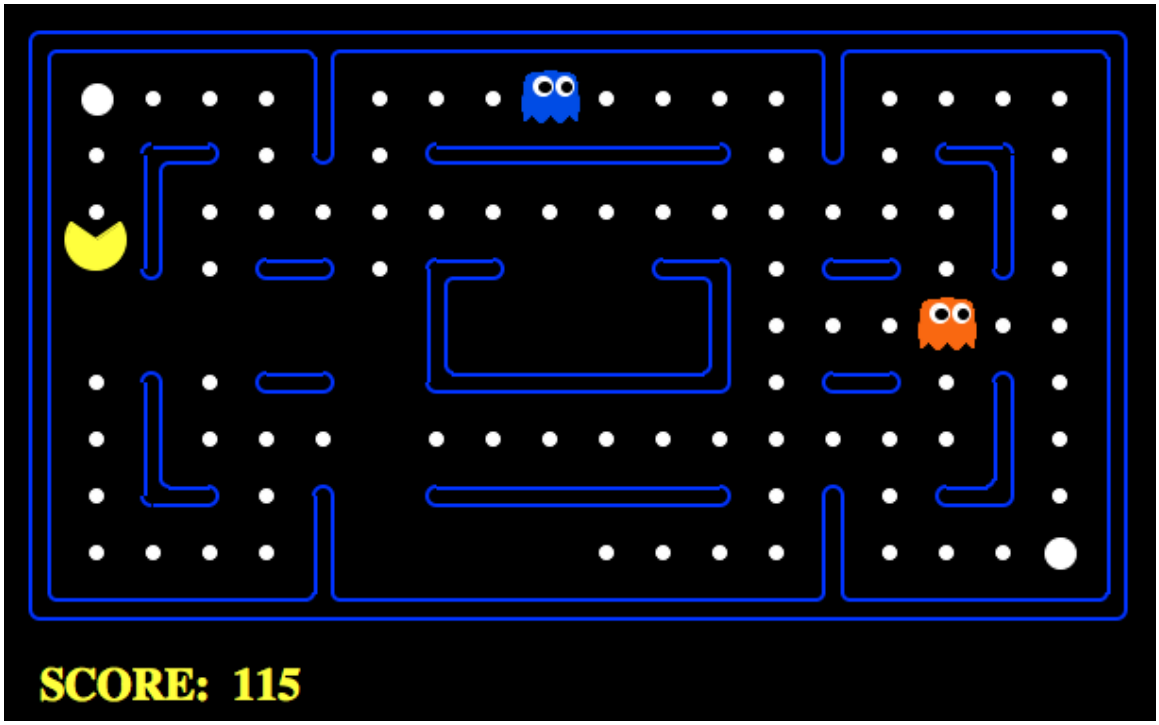


Figure 19: Medium sized classic Pacman map.

1. Problem Specification

Pacman requires the controller to select actions to avoid the ghosts while consuming dots. We can describe the environment by defining the following indices,

- t, turn or time, $t = \{1...T\}$, where T is the maximum time allowed to reach a waypoint, constant at T=1000 steps
- i, dots, $i = \{1..n\}$, where n is the number of dots that must be consumed,
- j, action index, $j = \{0, 1, 2, 3\}$,
- k, ghosts, $k = \{0, 1\}$,

and the following additional terms,

- p_t, the Pacman's position vector in two dimensional space at time t,
- $g_{k,t}$, the k^{th} ghost's position vector in two dimensional space at time t,
- o_t, the Pacman's orientation at time t,
- $c_{i,t}$, an indicator describing the state of the i^{th} dot at time t
- f_t, the number of dots remaining at time t.

2. RL Formulation

We define the RL control problem by the following tuple, $\langle \overrightarrow{s} \in S, a \in A, P_a^{s,s'}, R(s) \rangle$. The state description vector,s_t, $\overrightarrow{s_t} = \langle F_{d,t}, D_t \rangle$, describes the state of the pacman at time t, where $F(p_t, c_{i,t})_{d,t} \equiv$ the presence of food in the adjacent locations, $d \in (E, W, N, S)$ and $D(p_t, g_{k,t})_t \equiv$ the minimum distance to the nearest ghost.

$$F(p_t, c_{i,t})_{d,t} = \begin{cases} 0 & \text{if food not present} \\ 1 & \text{if food present} \end{cases}$$

$$D(p_t, g_{k,t})_t = \begin{cases} 0 & dist(p_t, g_{k,t}) < 6.0 \\ 1 & 6.0 < dist(p_t, g_{k,t}) \leqslant 15.0 \\ 2 & 15.0 \leqslant dist(p_t, g_{k,t}) \end{cases}$$

The reward function is defined,

$$R(s_t) = f_{t-1} - f_t \tag{40}$$

providing a graduated reward structure based on the number of dots consumed since our goal in this case is to clear the board of food, with a large negative penalty, -500 when awarded when the Pacman is killed.

This formulation resulted in extremely slow learning and in order to speed learning and improve performance we opted to include additional domain knowledge in the form of top-level strategies based on the same state. This approach is similar to one taken previously in attempts to incorporate RL into the Pacman environment and emphasizes an important point regarding learning speed. In most applied cases, the controller needs to learn policies in certain key *decision points*. This is worth noting as the same problem is encountered in the representation of human decision makers. The three strategies were to *flee*, to *findFood*, or to pursue a *greedy*, see Algorithms 8, 9, and 10. These strategies became the action space for the learner and the task became to map states to strategies.

Algorithm 8 Flee

1: Identify direction that increases the distance from the nearest threat.
2: Select direction.
3: Select strategy with RL controller.

Algorithm 9 Greedy

1: Identify adjacent direction that contains food.
2: Select direction.
3: Select strategy with RL controller.

Algorithm 10 findFood

1: Identify direction that reduces the distance to the closest food cell.

2: Select direction.

3: Select strategy with RL controller.

3. Empirical Results

In order to gain insight into the performance of the algorithm policy pairs on the control task we conducted computational experiments comparing the performance of DQ-C, SARSA, Q-Learning, SARSA(λ), and Q(λ) when paired with ϵ-greedy and Boltzmann. We evaluated performance across a range of parameters for each algorithm and show results for the best performance observed for each algorithm. Each combination is allowed to train for 1000 games, with a consistent random seed used across all combinations as a variance reduction measure. The mean percentage of the board cleared across all games was used as a measure of performance. SARSA(λ) paired with either ϵ-greedy or Boltzmann significantly outperformed competing algorithm policy pairs and with the exception of SARSA(λ), Boltzmann paired algorithms consistently outperformed their ϵ-greedy counterparts, see Figure 20.

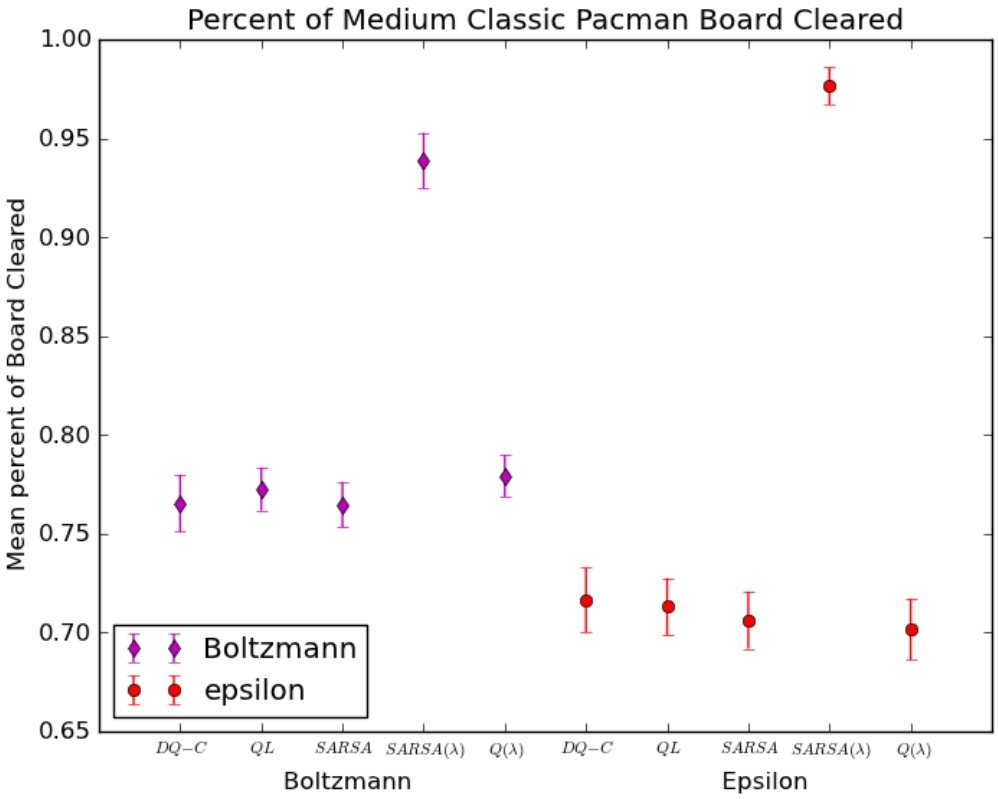


Figure 20: Mean percent of board cleared by all algorithm and policy pairs for best performing parameter settings.

4. Insights from PACMAN

In this section we examined the application of RL to a more complex domain with a challenging task, the control of the Pacman in the arcade game of the same name. This is a challenging adversarial environment with a potentially very large state space. We observed cases in this domain where DQ-C did not strictly outperform its competitors and a case where SARSA(λ) performed well. In all cases, learning was improved through the incorporation of domain knowledge through the use of strategy combinations. Here we are not asking the controller to learn atomic-level information, only the mapping between the

state of the controller and the discrete strategy choices. Note that the reward structure in this case still possessed the large delayed associated with the attainment of the waypoint in the previous section, but because of the difficulty in achieving the reward, which required complete clearance of the board, the controller in most cases never achieved it. The adversarial nature of the environment also served to shorten learning periods requiring multiple restarts. This aspect of the environment is similar in nature to challenges that would be present in the use of RL in a combat model, where attrition due to combat would create new states that would require additional learning-time.

E. INSIGHTS ON DQ-C

DQ-C showed a significantly better performance on standard benchmark problems and game-based domains than dominant algorithms in that rely on noisy hill-climbing processes. We make use of all available information by using the long-term historical reward to determine the value of the state-action pair at the time it was chosen and then ensure that we are weighting the most recent attempts of a state-action pair more heavily than early attempts through the use of an exponentially weighted average tied to continuous system time rather than simple counters. The use of discounted sums to develop an estimate of the long-term value of each attempt of a state-action pair linked to continuous environment time is novel as is the use of an exponentially weighted average tied to environment time. Weights are constantly adjusted with each attempt of a state action pair. We avoid temporal differencing and reduce the number of tunable parameters required to incorporate the notion of a recency bias from 3 to 2. In the next chapter we discuss the application of DQ-C to three applied DoD simulation use-cases.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. APPLICATIONS

This chapter details the incorporation of RL in a variety of applications, with a focus on comparison of Direct-Q Computation with similar algorithms. In the subsequent sections we provide example application of the algorithm in a two game-based domains and three DoD simulation models each, each related to a different practical use case. In the First section, RL is used to address two cases not currently covered by the Assignment and Scheduling Capability for Unmanned Aerial Vehicles (ASC-U) tool. The Second section shows a proof of principle integration of RL into COMBATXXI as a mechanism to control entity level decision making. The final section provides the results of the application of RL to verify the reward structure of a training simulation, UrbanSim.

A. UNMANNED AERIAL VEHICLE ASSIGNMENT AND SCHEDULING PROBLEM

Dynamic programming provides an appealing alternative to solve many applied military planning problems, such as the unmanned vehicle assignment and scheduling problem, that do not lend themselves to pure mathematical programming techniques. This problem includes dynamic UAV assignment constrained by the location of launch and recovery sites (LRS), and mobile ground control stations (GCS) with finite control capacity from which UAVs must be controlled throughout flight operations. Value in this problem is maximized by the assignment of UAV packages to mission areas with mission requirements that match package types. The Assignment Scheduling Capability for Unmanned Aerial Vehicles (ASC-U) simulation model provides an approximate solution to this problem using concepts from discrete event simulation and optimization to implement an a dynamic programming solution for tactical-level scenarios (Ahner et al., 2006; A. Buss, 2009).

1. Assignment Scheduling Capability for UAVs

ASC-U uses a discrete event simulation model coupled with the optimization of a linear object function to develop a feasible schedule for UAV mission assignments to mission areas. Value is obtained from the correct pairing of mission areas with UAV package types. UAV packages possess capabilities required to obtain value from a given mission area assignment based on the demands associated with that mission area. ASC-U provides a feasible solution at set intervals to a finite horizon assignment problem given complete information concerning the mission demands and associated values within the finite-time horizon. The simulation calculates an overall mission value for each assignment by considering the required flight time to each mission area and the amount of time each UAV covers the area. Note that all mission areas and their associated values are fully observable by the scheduler at the planning point for the entire time period being scheduled (Ahner et al., 2006). The scheduler provides each UAV only the initial mission area assignment, ensuring that only one UAV is assigned per mission area, and relies on a greedy heuristic to allow each UAV platform to select a subsequent mission should the mission area they are currently covering close. The need exists to improve the ability of the scheduler agent to develop near optimal plans schedules in cases with delayed and noisy rewards.

2. Problem Specification

The full dynamic programming formulation, see Appendix F, presented challenges due to the large state and control space, so in order to overcome these issues the problem was reformulated to a linear optimization problem that is solved at set time intervals during the course of a discrete event simulation, see Appendix F.

3. RL Formulation and Empirical Results

In this section, we describe two approaches to the application of RL to address the UAV assignment and scheduling problem. The formulations differ in manner in which the decision problem is formulated and in the appellation of RL to the problem. We will refer to the existing ASC-U framework as the *base case*.

a. Case 1 Formulation

In the first case, we simply replace the greedy heuristic to reassign UAV's following their initial allocation with an RL formulation. As each platform completes its current mission it is presented with a set of open feasible mission areas to choose from and in the absence of options will return to its LRS. In the base case, the platform is provided complete and accurate information regarding the value of each alternative and it chooses the mission with the maximum value at time 0 following the execution of the base case linear program. This approach also does not account for delayed emergent rewards and will miss the opportunity to pursue a higher valued mission area that is observable at the beginning of the planning horizon, but is open at some point in the future.

In order to provide an alternative to the base case heuristic we can formulate the problem as a bandit problem from the perspective of each platform as it becomes available. The arms in this problem are the set of open feasible mission areas and the option to return to base or loiter. The state is defined by the platform and reward is obtained following completion of the mission.

b. Case 1 Empirical Results

In order to evaluate this case we set up a small benchmark scenario that produces the situation where we expect that the base case will miss the opportunity to achieve a delayed reward, see Figure 21. In order to illustrate this case, we provide a simple scenario with a single LRS, single GCS and single UAV assigned. Two mission areas are observable in the initial planning horizon. The first mission area is available from time 0 to time 3 and provides a value of 10 per time unit covered. The second mission area opens at time 3 and remains open through time 7 providing a value of 10 per time unit covered. The base case will schedule the coverage of the first mission using the linear program, once the emergent mission area becomes available it will switch over to cover the emergent target with its remaining capacity. The base case will consistently miss out on additional value from the emergent target, no matter how many times that situation is encountered, since it follows a purely greedy policy with no real notion of state.

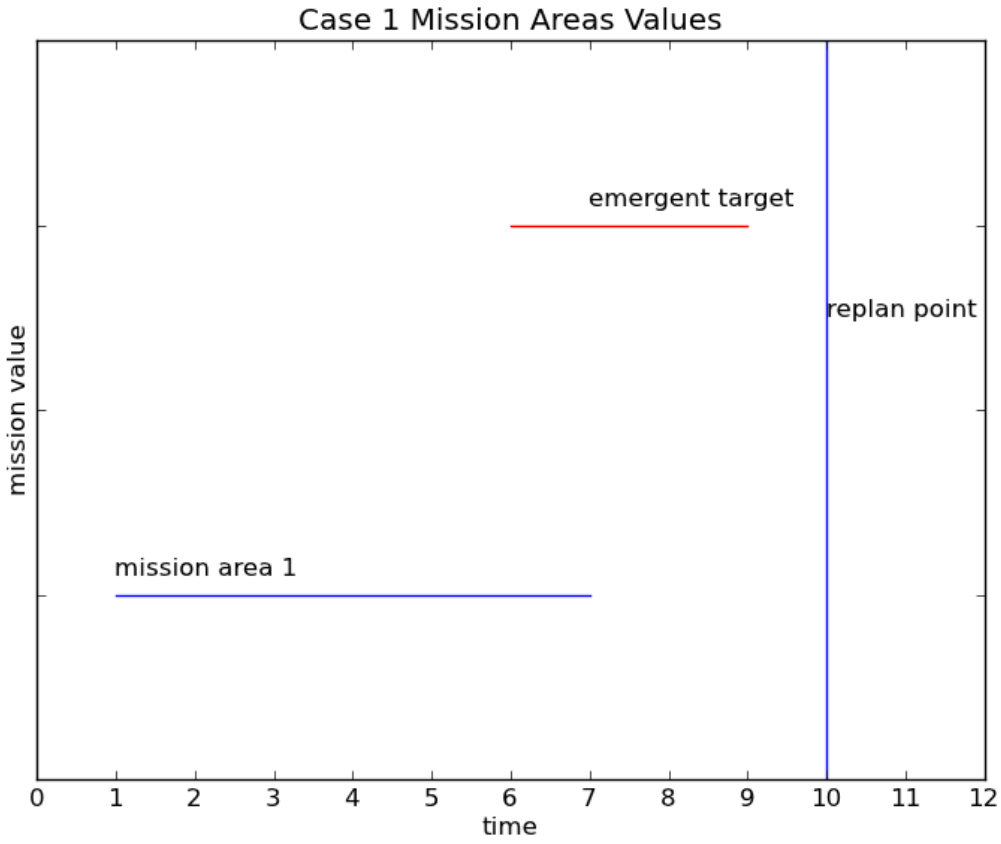


Figure 21: Case 1 mission area timing.

In the RL formulation we provide the alternative to simply preserve capacity in order to take advantage of a future high value mission area and the learning problem becomes one of identifying the appropriate time to preserve capacity versus behaving greedily, our familiar exploration and exploitation problem. This results in an improvement over the base case regardless of RL algorithm chosen, see Figure 22. In the course of a scenario this uncovered case serves to insert error into the schedule and since the scenario developer has no means of knowing how often this occurs during the course of a scenario

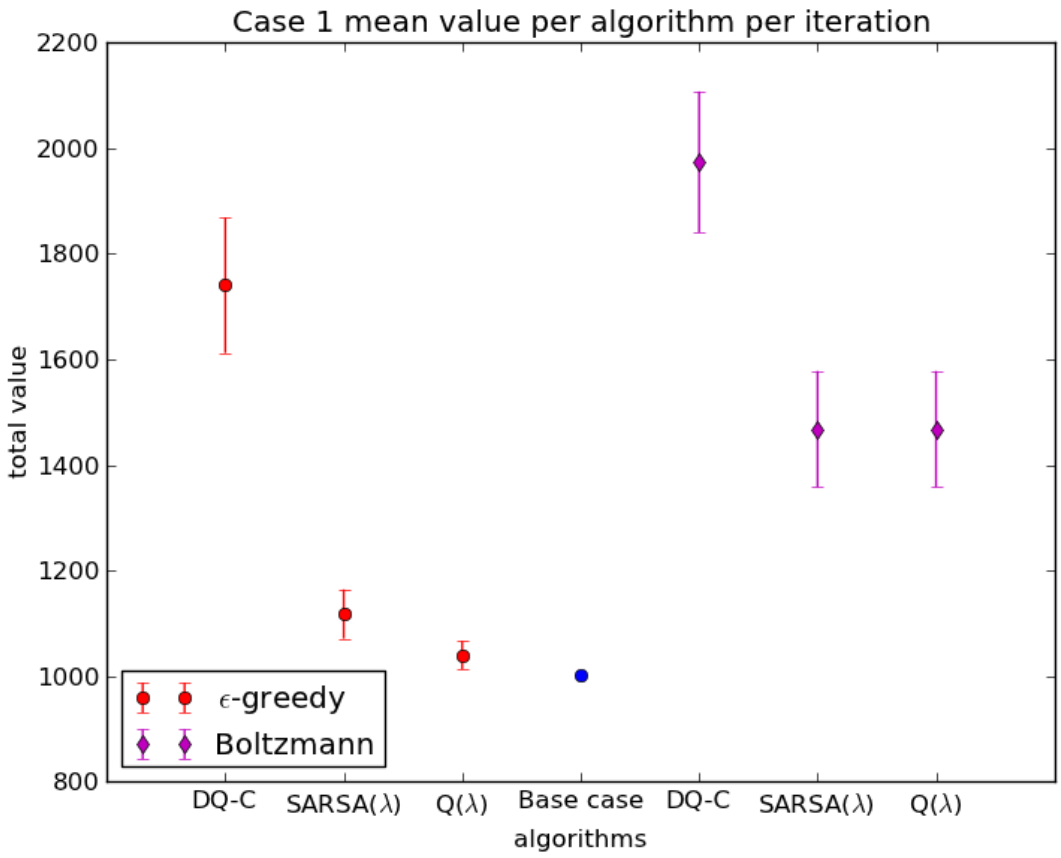


Figure 22: Case 1 results.

run, the potential for lost value, or regret, to accumulate over the course of a large scenario is great.

RL provides a mechanism to address this, but does require the use of multiple replications to arrive at a feasible solution, while the base case requires only a single run.

c. Case 2 Formulation

In the second case, we replace the linear program that provides the initial assignment with an RL formulation. The RL formulation is provided the same information

used by the base case at each planning interval. The set of available platforms are assigned to open mission areas. The base case assigns a only a single mission to each platform and does not see past the specified time horizon. We can expect that this approach will fail to achieve maximum value for those cases where all platforms are assigned to long durations mission areas at time t and new mission areas of higher value open at the next scheduled window.

In order to provide an alternative to the base case we provide the RL formulation with the same information provided to the base case and constrain it to decision making on the same interval, treating the assignment of each mission package to a mission area as an n-arm bandit problem, where the arms are defined by the mission areas and the option to "do nothing". State is defined by the mission package name, with the option of using the assignment time since the mission areas are defined by open and close times. In this case we will simply maintain the definition of state by the mission package name.

d. Case 2 Empirical Results

We first establish a scenario which reproduces the conditions where we would expect the base base to fail and confirm that the base case fails to obtain the maximum value under these conditions. In order to demonstrate this we establish a single LRS, with a single GCS and UAV assigned. The scenario consists of two mission areas, one open from time 0-9 which provides a value of 10 per time covered and a second open from time 11-15 with a value of 1000 per time covered, see Figure 23 and Table 7. The time horizon is set at 10, meaning that the base case will have full knowledge of mission demands out to time 10, and will optimize the mission package and mission area pairing at time 0 and time 10. The platform only has the ability to stay on station for 9 time units.

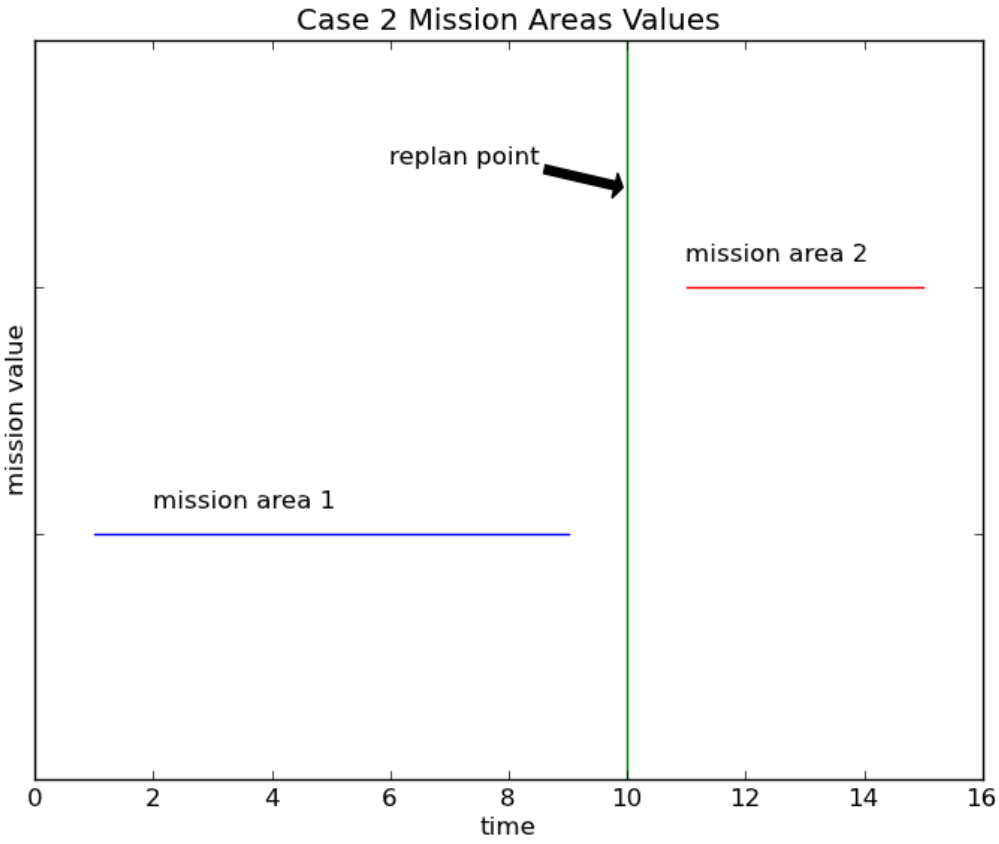


Figure 23: Case 2 mission area timing.

Table 7: Scenario configuration parameters for ASC-U Case 2.

PARAMETER	VALUE
OPTIMIZATION INTERVAL	10
SCENARIO LENGTH	20000
TOTAL MISSIONS	2000
PLATFORM TRANSITION TIME	1.5
OPERATIONAL ENDURANCE	9
TIME HORIZON	9

The base case consistently fails to cover the delayed reward available from the second open mission area since it develops an optimal schedule for the fully observable finite-time horizon problem with no regard to potential higher value mission areas just past that horizon. In this case, all resources are expended during the first planning period, leaving no free platforms to cover the second mission area. This case in particular occurred frequently enough in practice with larger scenarios to be noted by the developers.

The RL formulation is provided the same scenario and the same information regarding the state of the world as the base case, but is also provided the option to "do nothing" for each platform. The reward in this case is delayed since due to the scenario the value of the subsequent mission area is not available at the time of the subsequent decision period. The RL formulation is able to quickly identify the value of the delayed reward in this simple case. The base case and RL aggregate value scores using DQ-C and ϵ-greedy, see Figure 24, illustrate that the incorporation of RL can overcome the issues encountered by the base cases reliance on a fully observable environment. This emphasize the difference between a deterministic policy, such as a greedy algorithm or optimization, and a stochastic policy, such as those typically used in conjunction with RL.

We compare DQ-C's performance on this task with SARSA(λ), and Q(λ) when paired with ϵ-greedy, see Figure 24.

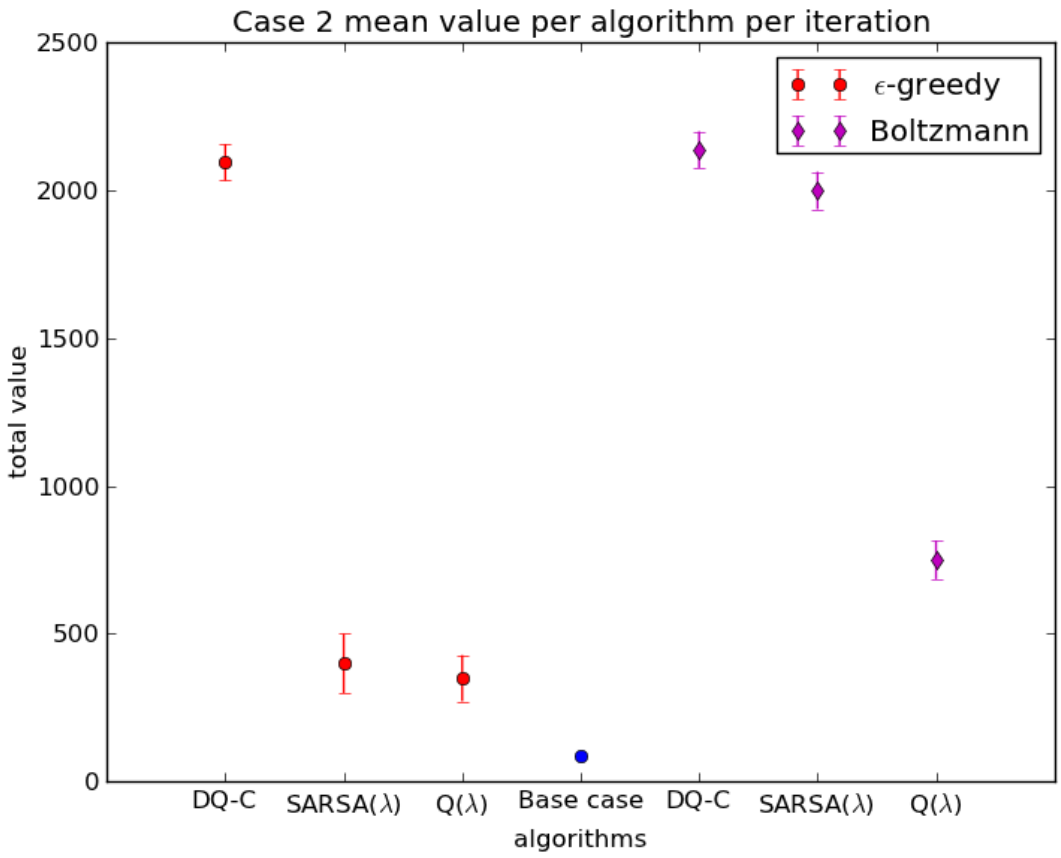


Figure 24: Case 2 comparison mean value per iteration for each algorithm.

4. Insights on the Use of Reinforcement Learning in a Scheduling Tool

The main insight in the application of RL within a scheduling tool is that RL does not require a fully observable environment as is the case in most optimization-based schedulers, such as ASC-U. Since RL does not make use of a deterministic policy, such as a greedy approach, RL is less likely make a consistent sub-optimal decision as we saw in the base case analysis of case 2. However, it does mean that RL will require more computational time and more iterations over the problem than the deterministic case. Depending on the use case and size of the problem the use of RL might not be practical, but it does

provide an approach to avoid those situations where a deterministic scheduler will consistently fail to achieve an optimal schedule. In the case of ASC-U, the case in question had been raised to the developers and was a known problem, but not one that a scenario developer could easily check for or avoid. This research contributes an approach to overcome this deficiency. As ASC-U moves toward scenarios with noisy rewards and only partially observable environments, the use of methods such as RL, which are designed to address these issues will be essential. Some key insights:

- RL methods are able to provide a solution that out performs the base case linear program in those cases where the optimal policy involves preserving capacity for delayed rewards not fully observable at the time of planning.
- DQ-C demonstrates the ability to quickly learn in cases with delayed rewards, but in cases where feedback is more immediate and where there is limited noise on the reward signal, other techniques perform as well as DQ-C.
- In this setting the use of a greedy-in-the limit approach to exploration and exploitation is appropriate, with the final state-action values serving as the scheduling policy.

In the next section, we will explore the application of RL algorithms to represent human decision making in COMBATXXI.

B. ADAPTIVE BEHAVIOR IN COMBATXXI

COMBATXXI is an entity level high resolution, closed-form, discrete event simulation used for analysis of future capabilities at the brigade level and below by the U.S. Army and the U.S.M.C. analytic organizations, TRADOC Analysis Center and the Operations Analysis Division respectively. COMBATXXI represents ground combat of light and heavy forces, air mobile forces, future forces, aerial assets, amphibious and urban operations. COMBATXXI additionally represents the communications network and the flow

of information on the battlefield. COMBATXXI has seen extensive use in the support of analysis of alternatives (AoA) for both the USA and USMC (Kunde, 2005).

Agents in analytic combat simulations, such as COMBATXXI, have relatively brittle decision making capability making, as illustrated in Figure 25, the development of scenarios time-consuming and making it challenging to conduct analysis of topics related to the value of information. The typical modeling and simulation agent requires an apriori mapping from states to actions, as represented by the arrows in Figure 25. This requires the anticipation of each possible state and an explicit decision, regardless of the environment dynamics, of the action the agent is to take in each case.

Figure 25: Typical modeling and simulation agent.

The incorporation of autonomous agents that make use of reinforcement learning, see Figure 26, present one potential technique to overcome this brittleness problem while potentially reducing scenario development time and improving the analysis of capabilities intended to impact the situational awareness of decision-makers. In the case of the reinforcement learning agent, rather than script a set of rules in advance, the agent develops an estimate of the value of its actions set in each state encountered. No hard coding of rules is required. This approach relies on the dynamics of the environment in which the agent is operating to provide feedback on performance to update its estimate, and in that manner

can also potentially inform the verification of simulation models. If following some learning period, the agent's ranking of the potential actions available from a given state does not match what a subject matter expert or modeler believes should be the case based on their full knowledge of the environment, then potential issues with the environment might be uncovered that would otherwise go unnoticed.

In this section, we demonstrate the use of reinforcement learning to control the actions of agents within a two cases in COMBATXXI, both oriented toward route planning.

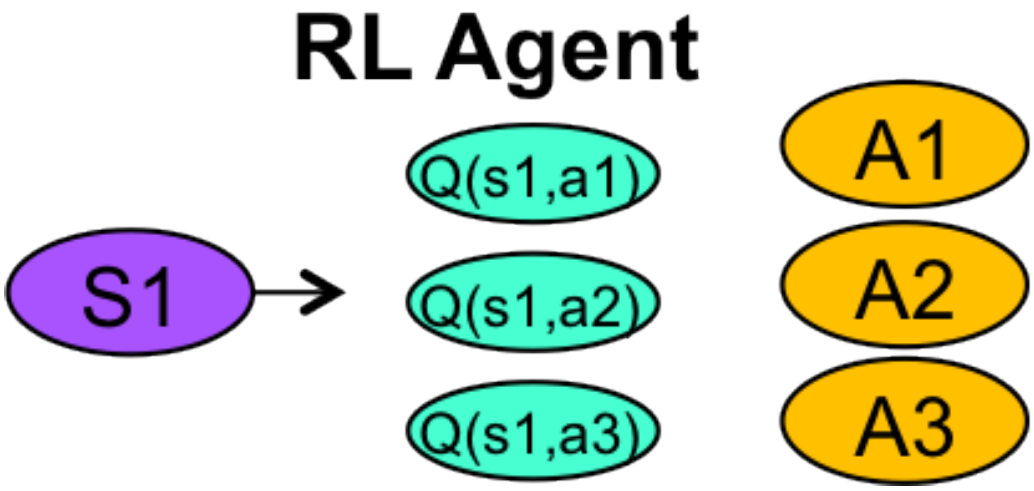


Figure 26: Modeling and simulation agent based on reinforcement learning.

1. Case 1 Formulation

In the first case, we examine a route selection problem in which the agent does not know the length of the route in advance. The state of the agent is defined by its waypoint location. The agent is provided a time based reward upon arrival at the final waypoint. Adaptive route selection is a common problem in combat simulations. In the real-world a unit will learn over-time which patrol route can be traversed in the shortest amount of time and with the least amount of threat. In this very simple case, the learning task to to learn which of two routes provides the shortest travel time to a destination see Figure 27.

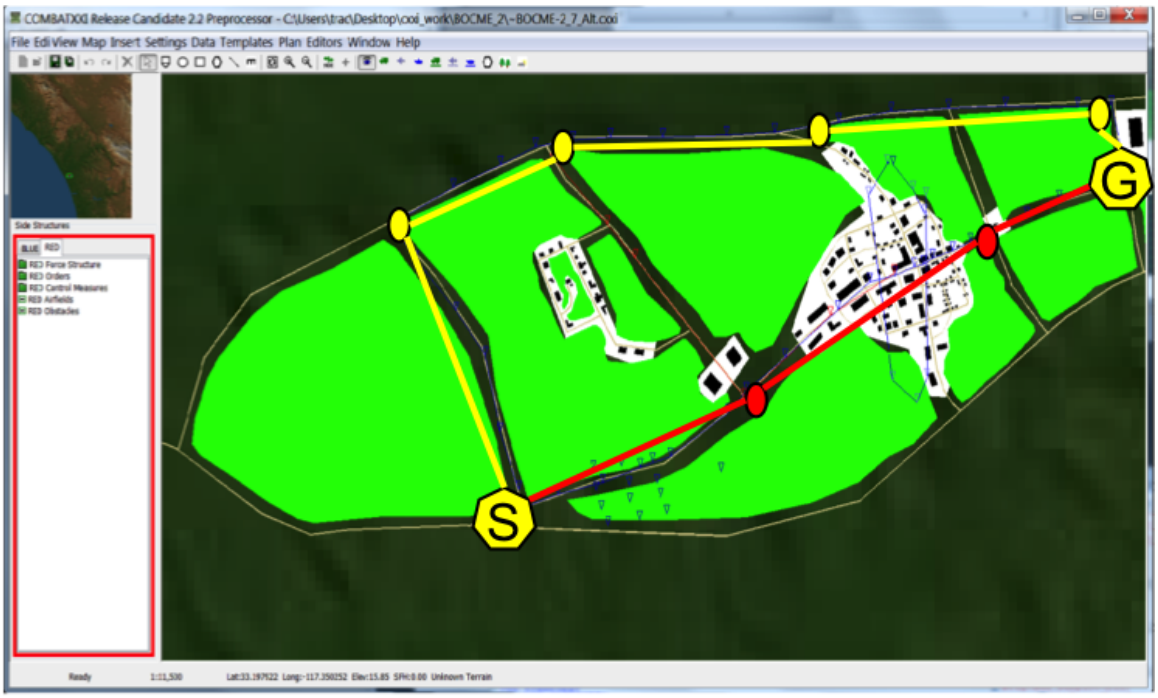


Figure 27: Route selection scenario COMBATXXI.

2. Case 1 Empirical Results

We see in this case that over a fairly short period of time, the agent learns to select the shorter of the two routes. DQ-C and SARSA(λ) both minimize regret and score well in total utility, see Figure 28. This case is a delayed reward case and it is interesting to note that Q(λ) does not perform well in comparison to DQ-C and SARSA(λ).

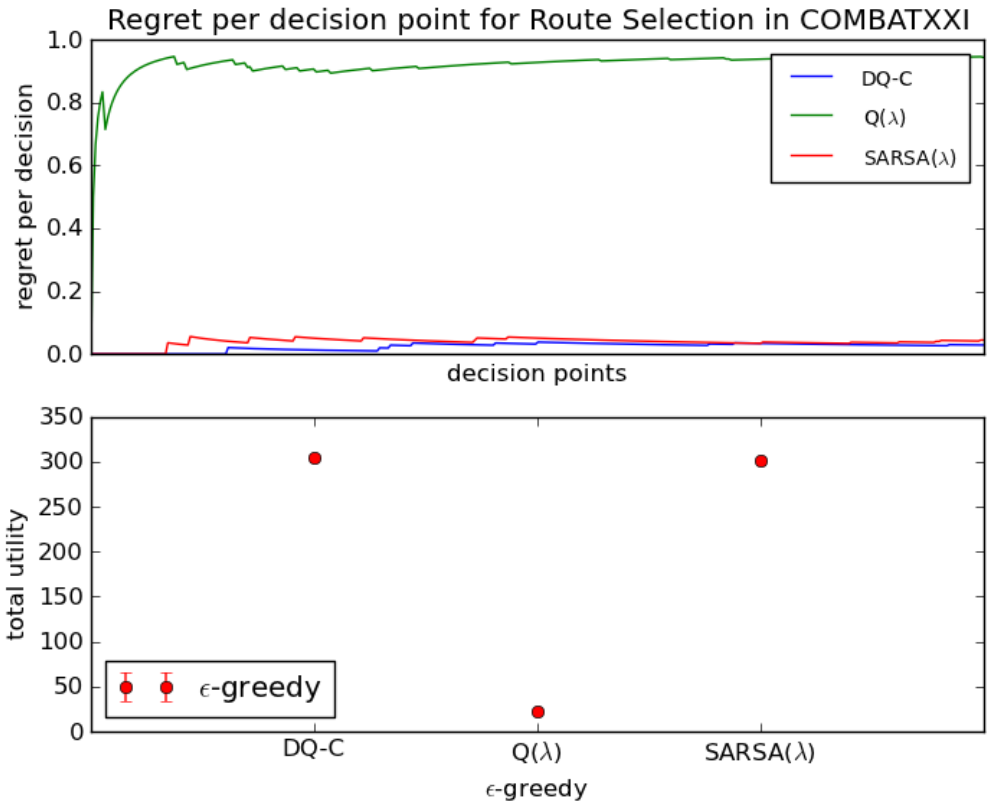


Figure 28: Route selection scenario COMBATXXI.

3. Case 2 Formulation

In the second case, we incorporate RL within a hierarchical task network designed to select a formation type based on information from the terrain, see Figure 29. At each waypoint, the agent makes a decision regarding the appropriate movement formation to use to move to the next waypoint. Reward is provided upon the arrival at the next waypoint based on the choice of movement formation. The task is to learn to use a bounding overwatch if there are more than two buildings between the start and end points of the movement leg. Decision-making often is informed by domain knowledge. In this case, we leverage hierarchical task networks to enable the agent to focus learning on only the rele-

vant decisions. The RL controller is incorporated into a decision node within the "move in formation" HTN. When presented with a decision situation, agent makes a decision based on the state of the world as defined by the number of buildings between it and the next waypoint. This is a simple case, but both the decision and the feedback could be made more complex in future cases.

Figure 29: Formation selection scenario COMBATXXI.

4. Case 2 Empirical Results

In this use case, we structure the reward to reinforce the desired TTP for a given situation. The state is defined by the number of buildings on the next leg of the circuit and the decision-making agents task is to learn the correct movement formation for a given number of buildings. A reward is provided for choosing the correct movement formation when the number of buildings is greater than 5 and a penalty of -1 is applied for choosing the incorrect movement formation when the number of building is less than 5. We compare the learning-time under this formulation for DQ-C, SARSA(λ), and Q(λ) in Figure 30. The results are consistent with previous cases, with DQ-C showing less regret per turn and a higher overall utility than the comparison cases. Note that this case does not involve a delayed reward and we see Q(λ) and SARSA(λ) switch places in the ranking.

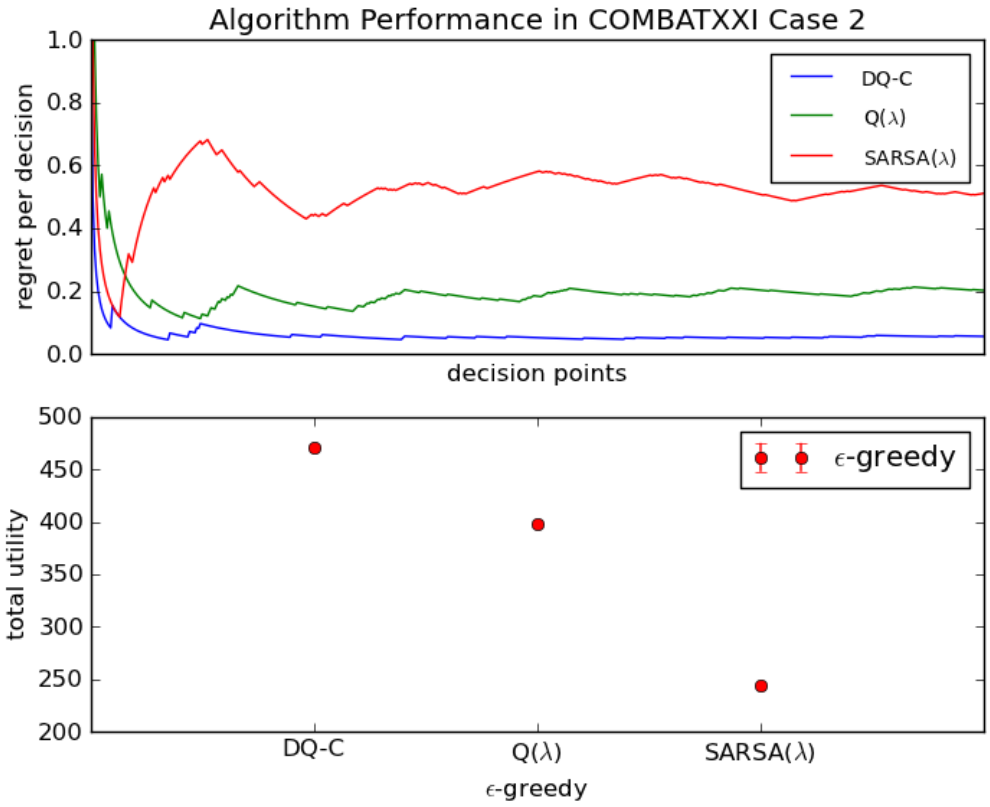


Figure 30: Formation selection scenario COMBATXXI.

5. Insights on the Use of Reinforcement Learning in Combat Simulations

The application of reinforcement learning within combat simulations in order to represent human decision making presents unique challenges, particularly in the analysis domain. Strict verification and validation guidelines require that algorithmic implementations be built on a valid conceptual model. In this case, RL is uniquely suited in that regard, having been developed in the literature on the study of animal behavior and continued and built upon in the literature on operant conditioning, and in fact still being contributed today by the field of neuroscience. RL provides a simple framework upon which to build complex adaptive behaviors. A key insight into the application of RL in the human behavior

representation use case is that an RL agent, much like a human, will progress through a learning period much faster with quality practice focused on the relevant decision points defined by the context. In this regard, the agent's perception of its state, or situation, informs it when it is appropriate to learn and when it is appropriate to exercise a deterministic policy known in advance. The use of RL in combat simulations lends itself to judicious use in specific decision situations. In order for learning to occur during a run the situation must be encountered multiple times, which implies that for practical purposes agent's should be put through training scenarios with similar situations and the state-action values stored for initialization in record runs of the combat model. A natural effect that could be gained from this type of approach is a mechanism to represent the effects of various level of soldier training and experience on performance.

The use of RL in conjunction with domain knowledge is also critical for the combat simulation model use case. This domain knowledge assists in the recognition of the situation and the need for a decision. The hierarchical task networks explored here provide a natural construct to facilitate this type of behavior in COMBATXXI.

C. VERIFYING THE REWARD STRUCTURE IN TRAINING SIMULATION

This section will apply reinforcement learning to the problem of verifying that the reward structure in game-based training simulations supports stated learning objectives. Specifically, we will apply RL to UrbanSim, a game designed to teach the doctrinal strategy to tactical-level leaders operating in a counterinsurgency setting (Wansbury et al., 2010). Trainees make a decision at each game turn based on their perception of the state of the environment, see Figure 31 and receive a numeric reward signal following the completion of each turn displayed in the lower right portion of the interface. In order to verify that the training system rewards the trainee for adhering to doctrinal actions we will employ an RL learner. The game developers identified that the verification of all potential paths through the training system required the use of automated mechanisms (Wang et al., 2012).

Figure 31: UrbanSim player interface.

1. Problem Specification

UrbanSim requires the player to select actions maximize a numeric reward over a finite-time horizon. The player chooses from a raw list of actions for each agent for each turn, with an action space as large as 341 for some agents on each turn. The order in which actions are chosen matters within the context of the game. Initially, we provide a description of the problem that incorporates the use of domain knowledge. We gradually increase the difficulty of the formulation until it mirror the unconstrained problem faced by the human player. We can describe the environment by defining the following indices,

- t, turn, $t = \{1...T\}$, where T is the maximum number of game turns allowed, constant at T=15 turns
- i, action index, $j = \{c, h, b\}$,
- j, agents, $j = \{0, 1, ..., 11\}$,

and the following additional terms,

- $A_{j,i,t}$, the action of agent j at turn t,
- r_t, the score returned to the player at turn t.

Note that the actions chosen by the reinforcement learner are strategy choices. These result in a draw from a bin of equally likely actions associated with that strategy. These actions were binned by subject matter experts based on the strategy they were most closely associated with. There is variance between the impact on the score of actions within each bin, resulting in a noisy reward signal for each strategy choice.

2. RL Formulation and Empirical Results

In this section, we describe three formulations explored with UrbanSim. The formulations differ in the state information provided to the agent and the timing of the reward signal.

a. Case 1 Formulation

Initially, we can formulate the problem as a sequential sampling problem allowing our agent to make a single decision on strategy at the beginning of the game and then receiving a score following the completion of the game with no opportunity to adjust strategy. This reduces the problem to a single state problem with an action space consisting of all possible combinations of the three actions. In this case, the default behavior of the system is to allocate each strategy to the third of the game in which it is specified in the ordering of the characters. for example, a strategy of *"clear, hold, build"* will result in the

use of a *"clear"* strategy by all agent during the first third of the game, a *"hold"* strategy the second third, and a *"build"* strategy during the final third of the game. The actions associated with each strategy can also be partitioned as *"lethal, non-lethal, mixed"* and further as *"clearly correct, mixed"*. This results in 162 strategy combinations, or arms, that must be explored.

b. Case 1 Empirical Results

In order to gain insight into the true value of each arm we first simply ran each of the 162 strategy choices for 30 replications each, with a unique random seed for each of the 30 replications, see Figure 32 and Figure 33. The results clearly indicate that some strategies are more preferable than others and that indeed it is reasonable to formulate this as a noisy bandit problem.

Figure 32: Mean and standard error of the final score of a 15 turn game following 30 replications of each of the 27 strategy combinations.

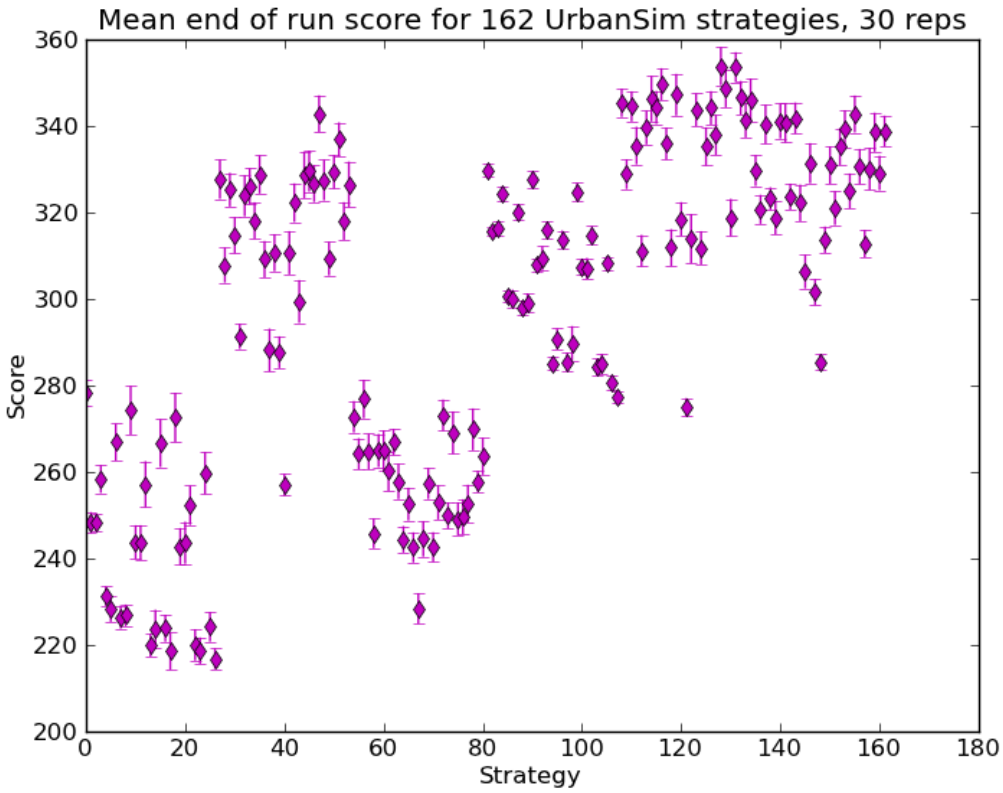


Figure 33: Mean and standard error of the final score of a 15 turn game following 30 replications of each of the 162 strategy combinations.

Table 8: Recommended strategies following 1000 training sessions by algorithm policy pair.

ALGORITHM-π	STRATEGY
DQ-C, BOLTZMANN	MKHBC
DQ-C, ϵ-GREEDY	SNHHB
Q(λ), BOLTZMANN	MRHHC
Q(λ), ϵ-GREEDY	MKBCC
SARSA(λ), BOLTZMANN	MRHHC
SARSA(λ), ϵ-GREEDY	MKBCC

We next formulated this as a 162 arm bandit problem for the purpose of understanding how long a learner might take to arrive at the optimal policy. We compare the results obtained using DQ-C, Q(λ), and SARSA(λ) paired with Boltzmann and ϵ-greedy strategies, see Figure 34. In this case we used the empirical maximum observed value as the best score in our regret calculations. Note that using this approach we could understand the learning-time required for a learner to arrive at the optimal policy by modeling this as a bandit problem outside of UrbanSim using the observed mean and variance of of each strategy as the reward on each arm. We see, Figure 34, that DQ-C paired with ϵ-greedy minimizes regret over the course of 1000 15 turn games with fixed parameters. Parameters used for each algorithm were set to the parameters which produced the best results in the noisy-bandit case.

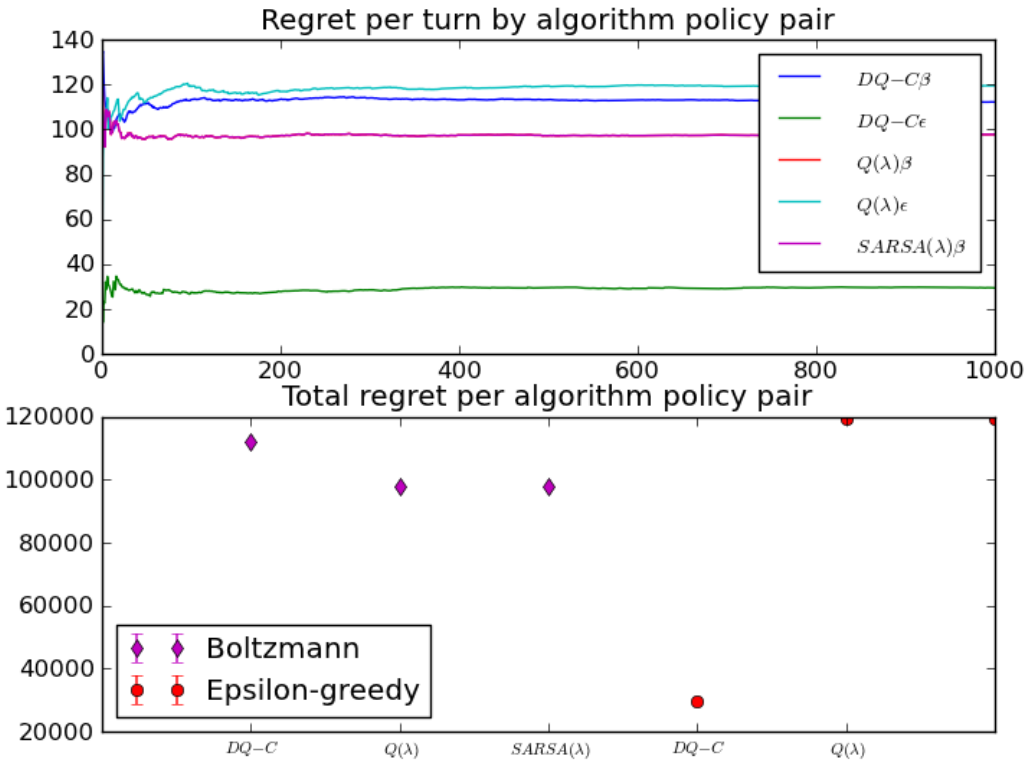


Figure 34: Regret per turn and mean total regret for 162-arm bandit formulation of UrbanSim constant exploration rate.

We also compare the performance of the algorithms using a decaying exploration rate, see Figure 35. The maximum score on this game is 600, the regret total regret and regret per turn for each pair are shown below following the completion of 500 games. Given the size of the action space we adopt a search then converge approach to control the level of exploration, with the remainder of the parameters tuned to the settings identified in our exploration of the noisy 10-arm bandit.

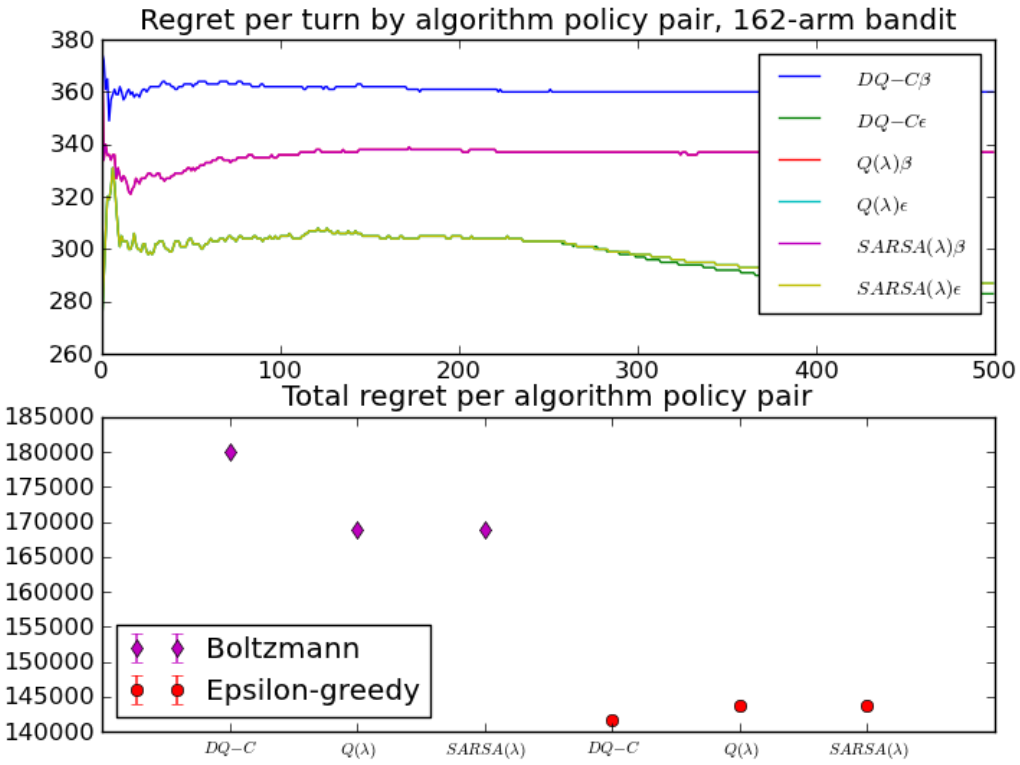


Figure 35: Regret per turn and mean total regret for 162-arm bandit formulation of UrbanSim using decaying exploration rate.

Table 9: Recommended strategies following 500 training sessions by algorithm policy pair using a decaying exploration strategy.

ALGORITHM-π	STRATEGY
DQ-C, BOLTZMANN	MKHBC
DQ-C, ϵ-GREEDY	SNBHH
Q(λ), BOLTZMANN	MRBCH
Q(λ), ϵ-GREEDY	SKCBB
SARSA(λ), BOLTZMANN	MRBCH
SARSA(λ), ϵ-GREEDY	SKCBB

We gain the following insights from this set of experiments,

- Formulating the simplified problem as a bandit allows us to effectively use RL in a sequential sampling role to adaptively adjust the design of experiments.
- When used in a sequential sampling setting to determine the optimal action choice the use of a high level of exploration is required initially to ensure broad sampling given a relatively small number of trials. While not a primary topic of this thesis, it is interesting to note that with only 500 replications we are able to identify a strategy with a value of 320 only 30 points below the max observed using batch runs with consumed a total of 4860 runs.

c. Case 2 Formulation

We can gain insight into the optimal policy for each agent each turn by incorporating the this information into the state description. Here we define $S = \langle \mathtt{agent}, \mathtt{turn} \rangle$. The action set consists of twelve possible strategy choices,

$$A = \langle \mathtt{mnc}, \mathtt{mnh}, \mathtt{mnb}, \mathtt{mkc}, \mathtt{mkh}, \mathtt{mkb}, \mathtt{snc}, \mathtt{snh}, \mathtt{snb}, \mathtt{skc}, \mathtt{skh}, \mathtt{skb} \rangle \tag{41}$$

As previously described, each strategy specifies a bin of actions that are categorized according to the three descriptors. In this case we use the following bin descriptions.

- m/s, access to either the full set of actions described by the following to categories or only those binned as *smart*
- n/k, access to either *kinetic* or *non-kinetic* bins
- c/h/b, access to either the *clear*, *hold* or *build* bin of actions

The reward function provides a numeric reward signal to the agent at the completion of each turn as in the case of a human player.

d. Case 2 Empirical Results

In this case we allow the agent to select actions by agent by turn, with a numeric reward following each turn with an initially high level of exploration that begins to decay after 500 games. Following the completion of 1000 games DQ-C provided the lowest end of game regret, see Figure 36.

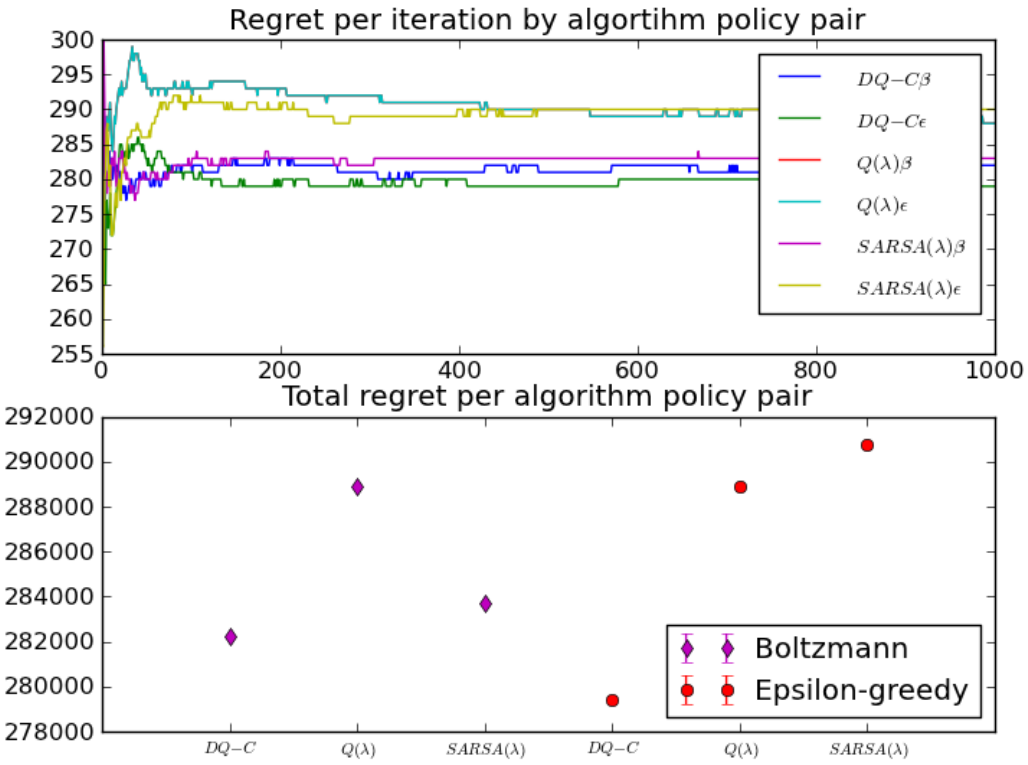


Figure 36: Regret per 15 turn game and total regret over 1000 games with decisions made by agent by turn.

DQ-C paired with ϵ-greedy provided the best empirical performance, followed by DQ-C paired with Boltzmann in this case. The size of the state space and frequent exploration likely contributed to Q(λ)'s relatively poor performance. SARSA(λ) paired with a Boltzmann strategy rounded out the top three, see Figure 37.

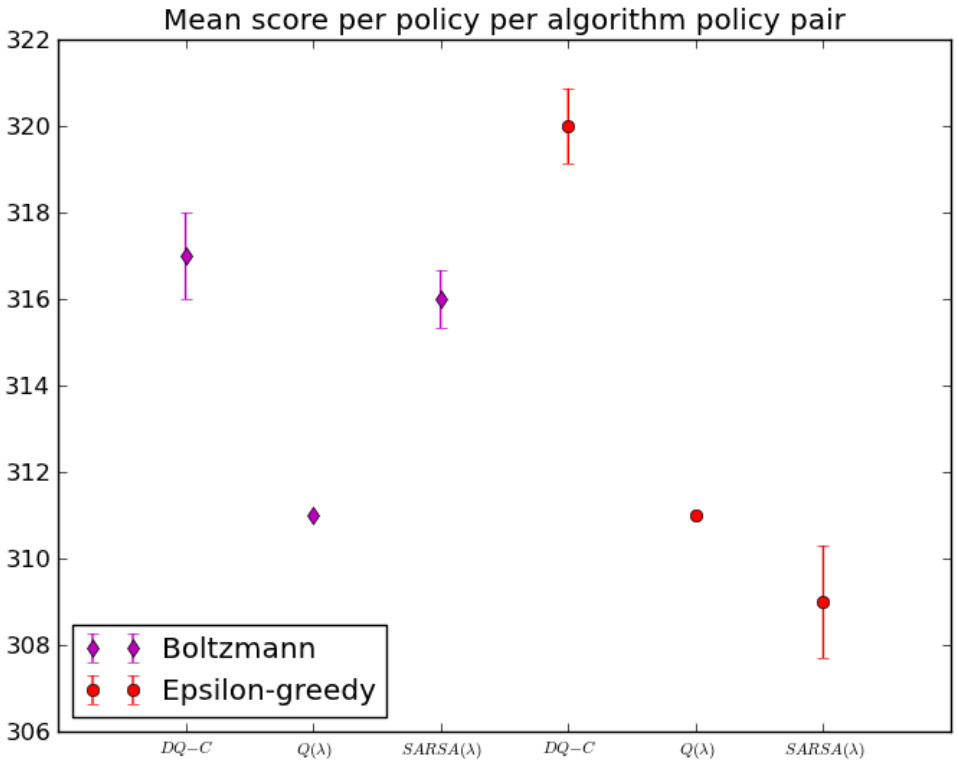


Figure 37: Mean score over 1000 games with decisions made by agent by turn for each algorithm policy pair.

Using information collected during the run we examine the value of each (s, a) as in the case of the battalion commander, see Table 10.

Table 10: Learned policy for Battalion Commander by turn for 15 turn UrbanSim game using DQ-C.

Strategy	1	2	3	4	5	6	7	8
SKC	9.4E-316	4.8E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
SKH	2.3E+02	2.4E+02	2.4E+02	2.4E+02	2.4E+02	2.4E+02	2.4E+02	2.5E+02
SKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
SNC	2.4E+02	2.4E+02	2.3E+02	2.3E+02	2.4E+02	2.4E+02	2.4E+02	2.4E+02
SNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
SNB	2.3E+02	2.6E+02	2.3E+02	2.4E+02	2.3E+02	2.4E+02	2.4E+02	2.5E+02
MKC	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
MKH	2.6E+02	2.3E+02	2.6E+02	2.4E+02	2.4E+02	2.4E+02	2.4E+02	2.3E+02
MKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
MNC	2.4E+02	2.4E+02	2.4E+02	2.6E+02	2.3E+02	2.4E+02	2.4E+02	2.4E+02
MNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
MNB	0.0E+00 9	2.4E+02 10	2.3E+02 11	2.3E+02 12	2.4E+02 13	2.4E+02 14	2.4E+02 15	2.4E+02
SKC	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
SKH	2.4E+02	2.5E+02	2.5E+02	2.6E+02	2.5E+02	2.6E+02	2.6E+02	
SKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
SNC	2.5E+02	2.5E+02	2.6E+02	2.6E+02	2.6E+02	2.6E+02	3.0E+02	
SNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
SNB	2.4E+02	2.5E+02	2.6E+02	2.9E+02	2.5E+02	2.6E+02	2.6E+02	
MKC	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
MKH	2.5E+02	2.4E+02	2.9E+02	2.6E+02	2.6E+02	2.6E+02	2.6E+02	
MKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
MNC	2.5E+02	2.8E+02	2.5E+02	2.6E+02	2.6E+02	2.6E+02	2.6E+02	
MNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
MNB	2.5E+02	2.5E+02	2.5E+02	2.5E+02	2.6E+02	0.0E+00	2.6E+02	

We can use this information to develop a policy matrix for each algorithm and exploration strategy, see Appendix G and Appendix H, employed that reflects the application of a greedy strategy following 1000 15 turn games. This policy matrix provides the recommended strategy for each agent for each game turn, see Table 11 and 12. Note that each algorithm exploration policy pair produces a slightly different matrix, reflective of the difference in the mean scores of each algorithm.

Table 11: Learned policy for all agents by turn for 15 turn UrbanSim game using DQ-C, ϵ-greedy.

STRATEGY	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
BATTALION COMMANDER	MKH	SNB	MKH	MNC	SKC	MNH	MNH	SNH	SKB	MNC	MKH	SNB	SNH	MNH	SNC
CA UNIT	SNC	MKH	MKH	MNH	MKB	MKB	SNC	SNH	MNC	SKH	SNH	MNH	SKC	MKH	SNH
E CO A	SKH	SKC	SKH	MKC	MNB	SKB	MNC	MNH	MNB	MKC	MNC	MKH	MNH	MKB	SKH
E CO B	SNB	MKB	SNC	MKC	SNH	SNB	MKB	SKB	SNC	SNC	SNB	SKH	MNH	MKB	MNB
F CO A	SNB	SKC	SNH	SKC	SKB	SKB	SNC	SNB	MNH	MNH	MNC	SNC	MNC	SKH	MNC
F CO B	SNH	SKH	SKC	MKB	SNC	SKC	MKB	SNC	SNC	MNC	MKB	SNC	SNH	MKC	MKB
G CO A	SNB	SKB	MNC	MKB	SNH	SKB	MNH	SNH	MKC	SNH	SKB	SKC	SKC	MNC	SNB
G CO B	SNB	MNC	SNB	MNH	SNH	MKC	SKC	MNH	SKH	MKC	SNB	SKC	SKC	SNH	SNH
H CO A	SKC	MNH	MNH	SNB	SKH	SKB	SKH	SNB	SKB	MKH	SKC	SNB	SKC	SKH	SNC
H CO B	SNH	MNB	SNB	MKB	SKH	MKB	SKC	MNC	SNB	SKC	SNB	MNB	SKH	SKH	SNH
QRF	SNH	SKH	MNH	MKC	MNB	MKH	MNB	SKH	SKC	SNH	SKH	MNH	MKH	MNH	SKB

Table 12: Learned policy for all agents by turn for 15 turn UrbanSim game using DQ-C, Boltzmann.

STRATEGY	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
BATTALION COMMANDER	MKH	MNH	MNH	SKC	MKH	MKB	MKB	MNB	SNH	SKB	MNB	SKC	MKB	SKC	SNC
CA UNIT	MKB	SNC	MNB	SKC	MKB	MKH	SNB	MNH	SNH	SKC	MNH	SNC	SKC	SNH	MNH
E CO A	SNH	SKH	MKH	SNC	SKB	MKC	MNC	SKB	MKC	MNH	MKB	MNH	SNC	SNC	MKB
E CO B	MNB	SNH	MKC	SNC	MKH	SKC	SNB	MNH	SNH	SNB	SKC	MNH	SKB	MNC	SNC
F CO A	MNH	MNC	SKC	MKC	SNB	SKB	SKC	MKH	SNB	SNB	SNB	SNH	SNC	MKC	MNB
F CO B	MNH	MKB	SNB	SNC	MKH	SNH	MKH	MKH	SKH	MKH	SNH	SNC	MKB	MKC	MKH
G CO A	SNH	MKB	SNC	MNH	SNH	MNC	SKH	MNB	SKC	MNB	MKC	MNC	SNH	SKB	SNC
G CO B	MNH	SNC	MKB	SNC	MKB	MKB	SKB	SNH	MKC	SNH	SNH	MKC	MKB	SKB	MKB
H CO A	MKH	SNC	SKC	MKC	MNB	MKC	SKC	MNB	MNH	MNH	SKB	SKB	MNH	SKB	MKB
H CO B	SNB	SKC	MKB	SKB	SKH	SKC	MNB	MNC	MNC	SKB	SNC	SKB	SNB	MNC	SKH
QRF	SNB	SKB	SNB	SKB	SNB	MNC	SNB	MNC	SKH	SKB	MNB	SKH	MKH	SNC	SKB

Using these policy tables, see Appendix H, a training developer could understand how well the reward signal from the training game supported the stated learning objectives.

e. Case 3 Formulation

We adjust the formulation in this case, defining the state by agent and turn as for case 2, but with the action space expanded to the full problem the human player faces. The learning problem consists of learning how to choose actions for 11 agents across 15

game turns. The action space for each agent is expanded from the 12 strategy choices observed in the previous case to an action space in excess of 300 for some agents. The action space for each of the agents is shown in Table 13

Table 13: Actions available by agent each turn

AGENT	ACTIONS PER TURN
BATTALION COMMANDER	42
CA UNIT	112
E CO A	341
E CO B	341
F CO A	341
F CO B	341
G CO A	341
G CO B	341
H CO A	341
H CO B	341
QRF	152

The reward function is the same as defined previously for case 3. For this case we use the algorithm policy pair from case 3 that provided the best performance, DQ-C paired with ϵ-greedy.

f. Case 3 Empirical Results

The RL system was configured to explore for the first 1000 games and then to begin reducing the ratio of exploration and exploitation for the next 300 games, see Figure 38. Initially, ϵ was set to 1.0 due to the large state-action space and was reduced following the completion of game 1000 like, $\epsilon = \frac{\epsilon_{initial}}{N_{games}-1000}$ for the remainder of the period with a final $\epsilon = \frac{1}{300}$. We produce a by agent by turn policy, see Table 14, for DQ-C

paired with ϵ-greedy only due to run time constraints. DQ-C paired with ϵ-greedy was chosen because of its previously discussed strong performance.

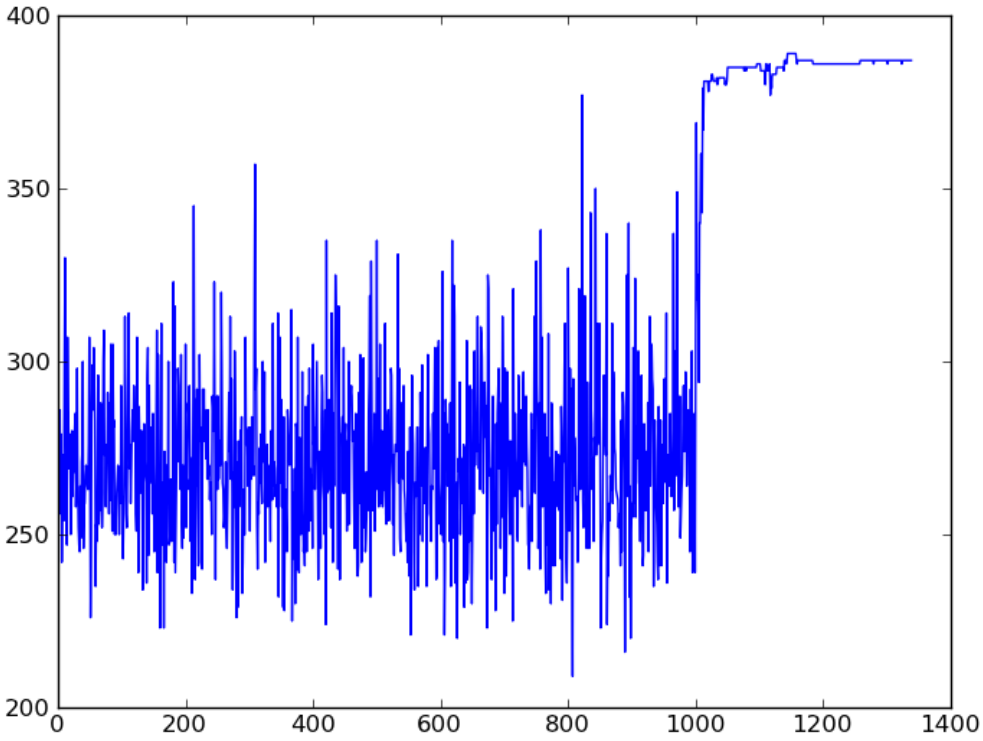


Figure 38: End of game score for 1000 learning games and 300 greedy games.

It is interesting to note that the final score from the greedy policy is higher than any previously observed during the learning period.

Table 14: Learned action policy for all agents by turn for 15 turn UrbanSim game using DQ-C, ϵ-greedy.

STRATEGY	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
BATTALION COMMANDER	13	5	12	8	31	36	41	3	40	29	35	25	8	33	15
CA UNIT	6	88	14	5	88	51	33	69	77	38	18	42	30	38	63
E CO A	178	312	131	32	152	261	57	71	234	299	236	235	318	266	230
E CO B	282	236	324	314	249	193	125	114	141	151	195	104	28	114	333
F CO A	322	212	16	325	320	258	60	133	114	162	200	245	253	317	290
F CO B	262	170	337	102	291	331	154	169	312	2	166	149	1	142	55
G CO A	321	233	254	169	77	46	35	46	187	31	233	93	28	73	57
G CO B	48	166	158	327	166	301	26	305	160	292	108	111	179	98	317
H CO A	252	294	139	290	180	27	177	286	297	109	338	276	42	224	236
H CO B	337	20	322	289	319	11	181	289	49	78	338	224	285	332	152
QRF	49	56	69	37	86	92	98	108	53	25	89	46	34	93	13

Using this table, see Table 14, scenario or game developers can examine the paths to ensure that the desired behaviors are being reinforced by the training system. We can then turn these index tables into the actual action descriptions shown chosen by a human player, see Tables 15,16,17,18,19.

Table 15: Learned actions for all agents from turn 1-3 in 15 turn UrbanSim game using DQ-C, ϵ-greedy.

Agent	1	2	3
Battalion Commander	Give Propaganda-Kasiriyah Quarter	Give Propaganda-Market District	Give Propaganda-Kasiriyah South Quarter
CA unit	Give Propaganda-Nahiyat Musalla	Repair-Kasiriyah Quarter School	Give Propaganda-Hisar Bakr
E CO A	Remove-Highway 5	Recruit Soldiers-Nahiyat Abu Qasim	Repair-Shipping Terminal Cement Plant
E CO B	Cordon and Knock-Northern Area	Patrol Neighborhood-Hisar Bakr	Recruit Soldiers-Kasiriyah South Quarter
F CO A	Recruit Soldiers-Nahiyat Artet	Attack Group-Shiite Death Squads	Arrest Person-Rushdi Kaliq
F CO B	Seize Structure-Granary 2 (IED Manufacturing Plant)	Remove-Minaret	Recruit Police-Nahiyat Artet
G CO A	Recruit Soldiers-Kassad Quarter	Patrol Neighborhood-Nahiyat Artet	Seize Structure-Mekel Bus Station
G CO B	Cordon and Search-Kasiriyah South Quarter	Remove-Adnan Mosque	Repair-Police Western HQ
H CO A	Seize Structure-Firing Range	Cordon and Knock-Kasiriyah Quarter	Repair-Firing Range
H CO B	Recruit Police-Nahiyat Artet	Arrest Person-Police Colonel Qasim Bashir	Recruit Soldiers-Nahiyat Artet
QRF	Remove-Nahiyat Artet Mosque	Remove-Kassad Water Storage Facility	Remove-Hwy 2 Gas Station

Table 16: Learned actions for all agents from turn 4-6 in 15 turn UrbanSim game using DQ-C, ϵ-greedy.

Agent	4	5	6
Battalion Commander	Give Propaganda-Hisar Kerhad	Host Meeting-DEP Mayor Dabir Ta'anari	Host Meeting-Mayor Anwar Sadiq
CA unit	Give Propaganda-Market District	Repair-Kasiriyah Quarter School	Host Meeting-Police Colonel Qasim Bashir
E CO a	Treat Wounds/Illnesses-Tribe 7	Repair-Hasim Road Textile Mill	Seize Structure-Omar Hasim's Office
E CO b	Recruit Soldiers-Malmoud Quarter	Seize Structure-Highway 5	Remove-Highway 2 Bridge
F CO a	Recruit Soldiers-Hisar Bakr	Recruit Soldiers-Hisar Sifiri	Seize Structure-City Municipal Complex
F CO b	Host Meeting-Abim Ubaya	Cordon and Knock-Kassad Quarter	Recruit Police-Market District
G CO a	Remove-Water Treatment Plant	Pay-Kurdish Shepherds	Cordon and Search-Kassad Quarter
G CO b	Recruit Police-Nahiyat Abu Qasim	Remove-Adnan Mosque	Set up Checkpoint-Market District
H CO a	Cordon and Knock-Hisar Sifiri	Remove-Yaminah Market	Arrest Person-Omar Hasim
H CO b	Cordon and Knock-Hisar Kerhad	Recruit Soldiers-Hisar Kerhad	Give Propaganda-Nahiyat Artet
QRF	Remove-Mekel Market	Patrol Neighborhood-Nahiyat Ayadh	Patrol Neighborhood-Hisar Sifiri

Table 17: Learned actions for all agents from turn 7-9 in 15 turn UrbanSim game using DQ-C, ϵ-greedy.

Agent	7	8	9
Battalion Commander	Give Propaganda-Nahiyat Ayadh	Advise-Police Colonel Qasim Bashir	Host Meeting-Rushdi Kaliq
CA unit	Support Politically-Terlan Demikan	Repair-Shipping Terminal Cement Plant	Joint Investigate-Mekel Quarter
E CO A	Dispatch Individual-Ebrahim Hafiz	Patrol Neighborhood-Kasiriyah South Quarter	Set up Checkpoint-Nahiyat Ayadh
E CO B	Support Politically-Rushdi Kaliq	Repair-Mekel Bus Station	Repair-Highway 2 Bridge
F CO A	Repair-Trash Depot	Support Politically-Rushdi Kaliq	Repair-Highway 2
F CO B	Remove-Water Treatment Plant	Recruit Soldiers-Nahiyat Abu Qasim	Give Propaganda-Malmoud Quarter
G CO A	Cordon and Search-Kassad Quarter	Remove-City Municipal Complex	Treat Wounds/Illnesses-Tribe 1
G CO B	Set up Checkpoint-Hisar Sifiri	Repair-Iraqi Army BDE HQ	Cordon and Knock-Nahiyat Artet
H CO A	Cordon and Knock-Market District	Set up Checkpoint-Nahiyat Abu Qasim	Host Meeting-Mayor Anwar Sadiq
H CO B	Cordon and Knock-Hisar Kerhad	Cordon and Search-Kasiriyah Quarter	Pay-Tribe 3
QRF	Seize Structure-Al-Hamra' Central School	Remove-Highway 2 Bridge	Dispatch Individual-Omar Hasim

Table 18: Learned actions for all agents from turn 10-12 in 15 turn UrbanSim game using DQ-C, ϵ-greedy.

Agent	10	11	12
Battalion Commander	Host Meeting-Rushdi Kaliq	Host Meeting-Ebrahim Hafiz	Give Gift-Ali Bakr Obeidi
CA unit	Joint Investigate-Mekel Quarter	Treat Wounds/Illnesses-Tribe 1	Joint Investigate-Nahiyat Artet
E CO A	Set up Checkpoint-Nahiyat Ayadh	Patrol Neighborhood-Hisar Bakr	Patrol Neighborhood-Kasiriyah Quarter
E CO B	Repair-Highway 2 Bridge	Remove-Al-Qassas Brigade Safehouse	Host Meeting-Jafar Udnan
F CO A	Repair-Highway 2	Remove-JAAS Safehouse	Seize Structure-Al-Hamra' City Hospital
F CO B	Give Propaganda-Malmoud Quarter	Remove-Adnan Mosque	Repair-Granary 2 (IED Manufacturing Plant)
G CO A	Treat Wounds/Illnesses-Tribe 1	Patrol Neighborhood-Nahiyat Artet	Joint Investigate-Hisar Kerhad
G CO B	Cordon and Knock-Nahiyat Artet	Host Meeting-Asad	Host Meeting-Terlan Demikan
H CO A	Host Meeting-Mayor Anwar Sadiq	Recruit Police-Kasiriyah South Quarter	Seize Structure-Sewage Processing Facility
H CO B	Pay-Tribe 3	Recruit Police-Kasiriyah South Quarter	Patrol Neighborhood-Malmoud Quarter
QRF	Dispatch Individual-Omar Hasim	Patrol Neighborhood-Nahiyat Musalla	Remove-Bakr District School

Table 19: Learned actions for all agents from turn 13-15 in 15 turn UrbanSim game using DQ-C, ϵ-greedy.

Agent	13	14	15
Battalion Commander	Give Propaganda-Hisar Kerhad	Host Meeting-Police Colonel Qasim Bashir	Give Gift-Fouad Sulimani
CA unit	Pay-Tribe 4	Joint Investigate-Mekel Quarter	Support Politically-Jafar Udnan
E CO A	Recruit Soldiers-Mekel Quarter	Seize Structure-Al-Qassas Brigade Safehouse	Patrol Neighborhood-Hisar Kerhad
E CO B	Treat Wounds/Illnesses-Kurdish Shepherds	Support Politically-Rushdi Kaliq	Recruit Police-Nahiyat Musalla
F CO A	Seize Structure-Transformer Station	Recruit Soldiers-Nahiyat Musalla	Cordon and Knock-Hisar Sifiri
F CO B	Give Propaganda-Nahiyat Abu Qasim	Repair-Kasiriyah Quarter School	Release Person-Jafar Udnan
G CO A	Treat Wounds/Illnesses-Kurdish Shepherds	Dispatch Individual-Mayor Anwar Sadiq	Release Person-Police CPT al-Nasser
G CO B	Remove-Artet District School	Joint Investigate-Kasiriyah Quarter	Recruit Soldiers-Nahiyat Musalla
H CO A	Cordon and Search-Nahiyat Musalla	Patrol Neighborhood-Malmoud Quarter	Patrol Neighborhood-Hisar Bakr
H CO B	Cordon and Knock-Hisar Adnan	Recruit Police-Nahiyat Musalla	Repair-Hasim Road Textile Mill
QRF	Remove-Al-Hamra' City Hospital	Patrol Neighborhood-Kassad Quarter	Dispatch Individual-Fouad Sulimani

3. Insights on the Verification of Training Simulations

One of the chief insights was the notion of treating the training simulation as a bandit problem. Viewing the problem in this manner, the challenge from the design standpoint is to specify the reward on each arm so as to correctly reinforce the behaviors associated with the learning objectives. Since the learning objective is at the strategy level, the mean value of the action bin associated with each strategy specifies the reward on each of the strategy arms. This is equivalent to a noisy bandit problem. Using this paradigm, the level of difficulty of the training simulation could be directly controlled by adjusting the noise on the reward signal. Following an easy-to-hard progression, the level of noise associated with

the strategy arms could start out at 0 and be incrementally adjusted based on the trainees progression, with a goal of trainees maintaining the optimal policy even in a noisy environment in the most difficult cases. The development of optimal polices for each agent could also support the use of semi-automated forces to allow trainees to focus on a single role in the game. This application areas is open to much future research that builds upon the results documented here and in the master's thesis of U.S. Army Major Brian Voght, whose research was supported by this effort.

D. CONCLUSIONS

In this section, we demonstrated novel applications of RL within three different use cases relevant to military modeling and simulation, the assignment and scheduling of unmanned systems, the representation of human decision making in a combat simulation, and in the verification of the reward structure in a training simulation.

In each case, the use of RL was unique and served to address known deficiencies or needs within the target simulation. In ASC-U, we observed that RL was could overcome specific case in which the finite horizon optimization approach failed. In COMBATXXI, we observed that the incorporation of RL in specific decision points where domain knowledge is available could provide adaptive behaviors more representative of human behavior and potentially represent the various training levels present in a real population. In UrbanSim, RL demonstrated its utility as an approach to address the challenge of verifying that the reward signal provided to a trainee was reinforcing the state learning objectives. In each case, the performance of DQ-C was competitive with the comparison algorithms.

- The definition of state and the size of the state space greatly impact the learning rate of RL techniques.
- The use of domain knowledge, as a human would, can greatly improve the performance of RL methods.

- The identification of the appropriate decisions that potentially require RL, such as decision nodes on a decision tree, can guide the effective use of RL.
- In application within combat simulation, the use of a training period to allow behaviors to form is likely required prior to the conduct of record analysis runs.
- The use of RL as a means of verifying the reward structure within training simulations is a natural fit for those cases where the verification does not lend itself to standard design of experiments techniques.

In the next chapter we develop a practical cognitive architecture for use in discrete event simulation that places RL into the broader context of human behavior models.

V. DEVELOPMENT OF A PRACTICAL COGNITIVE ARCHITECTURE

> We appeal to ordinary perception to arrive at our physical theories, yet those same theories seem to undermine that everyday perception, which is rich in meaning. - Bertrand Russell, Analysis of Matter (1927)

The purpose of this section is to review the literature and current state of the art regarding cognitive social simulations, cognitive architectures and their application in support of military decision making and analysis. Cognitive social simulation, a derivative of agent based social simulation, combines the use of cognitive architectures with traditional agent based social simulation approaches.

This section develops a practical cognitive architecture that places reinforcement learning within the broader context of human behavior models. We develop a conceptual model of the cognitive architecture and describe its implementation in a discrete event simulation framework. We also describe the application of reinforcement learning and the cognitive architecture within an agent based social simulation. Results reported in this section were previously discussed in several conference proceedings, an accepted in-press journal article (S. Papadopoulos et al., 2013), and student thesis work supported by this effort (J. K. Alt et al., 2011; Ozcan et al., 2011; Ozkan, 2011; M. Papadopoulos, 2010; Pollock et al., 2011; McKaughan, 2011).

A. GENERAL FRAMEWORK AND IMPLEMENTATION DESCRIPTION

The cognitive architecture proposed in this research provides a minimalist approach for modeling human decision making based on the concept of situation. While multiple cognitive architectures exist in the literature, the framework proposed here seeks to incorporate the impact on relevant concepts from cognitive science, psychology, and social psychology in a relatively simple manner. The intent is to avoid a kitchen sink approach by identifying a framework to account for the influence of these notions using the smallest

number of concepts and parameters possible. The prototype architecture provides a framework for experimentation with software agents for use in agent based social simulations with potential for the use of the architecture in conjunction with empirical data collection efforts. The need for an agent decision making architecture centered on the recognition of a given situation is highlighted by the literature on decision making and the need to reduce complex state spaces in agent environments (Klein, 1993; Russell & Norvig, 2010).

Agent architectures capable of recognizing relevant situations enable the use of algorithms such as RL (Sutton & Barto, 1998). RL provides multiple techniques to enable software agents to select actions in given situations based on a reward policy specified by the modeler. The use of utility based rewards allows these policies to be tailored to the desired use case and role (J. K. Alt et al., 2011).

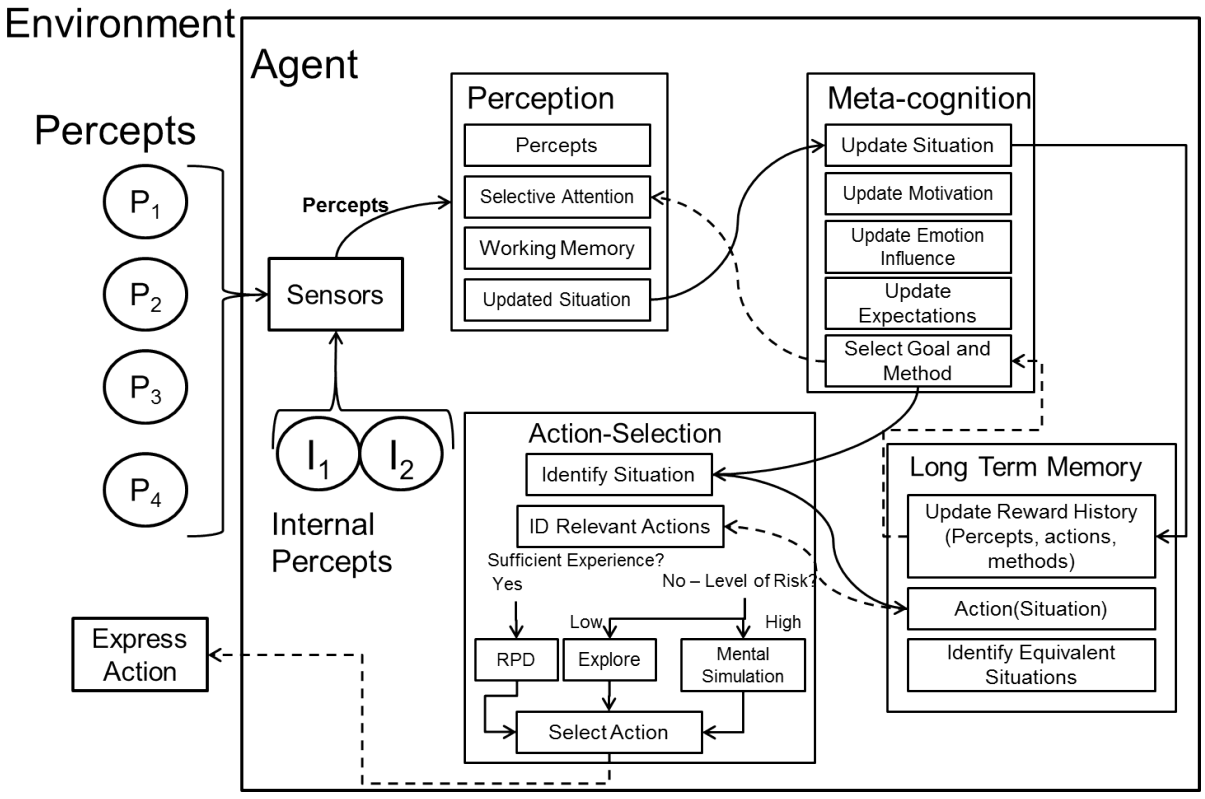


Figure 39: Practical cognitive architecture full conceptual model.

Applications such as battlefield command and control systems and agent based social simulations require agents capable of allocating selective attention to relevant percepts in a given context. This combination of bottom up and top down processing in conjunction with the constraints of working memory facilitate the construction of a situation. The framework allows for the representation of human behavioral phenomena such as change blindness, where changes in a scene are not observed due to the effects of top down processing on selective attention. This framework also provides a mechanism for agents to participate in collective learning within a social network, to determine which agents to communicate with and to determine what messages should be attended to.

This section describes the proposed cognitive architecture from a conceptual standpoint.

1. Perception

The main function of the perception module is to form a situation constrained by the limits of working memory and informed by selective attention. Percepts arrive to the perception module via sensors that sense information from the environment and from the internal agent feedback mechanisms. Note that this architecture treats inter-agent communications through the receipt of information via percepts and the decision to communicate via action selection.

Percepts are screened for relevance based on selective attention and if found relevant to the current situation are processed into working memory. Selective attention is driven by top down processing from the task and context (Anderson, 2005; Anderson & Schunn, 2005). Selective attention is influenced by the current motivations and emotions from the meta-cognition module, which serves to identify the goal. In communications selective attention is informed by information regarding the relationship with the other agent and notions such as trust. Working memory is limited to 7-10 percepts, the generally accepted limit (Wickens & Hollands, 2000). The final set of percepts is considered a description of the current situation, considering both external perceptual information and information from the agents internal state.

2. Meta-cognition

The meta-cognitive module provides the agents top-down direction based on motivations and emotions elicited by a given situation input from the perception module. Meta-cognition is broadly defined as any cognitive process that monitors or controls other aspects of cognition or thinking about thinking (Davis & Venkatamuni, 2010). Meta-cognition is described by Flavel as occurring in three phases: 1) meta-cognitive knowledge stores information regarding the environment, task, and known strategies; 2) meta-cognitive experience stores information describing previous means of achieving a given result; 3) meta-cognitive regulation describes the process of monitoring and controlling progress on cognitive tasks (Flavell, 1979).

The meta-cognitive module also hosts the agents information regarding the motivation of agents behavior. The input to the meta-cognition module is the most recent situation provided by the perception module. Using this updated situation the meta-cognition module conducts an update to determine which motivations are active and to assess the impact of the new situation on its goal state. The situation object in conjunction with information from long-term memory is used to form expectations about likely future situations (J. K. Alt et al., 2011). Goals and methods are selected using input on motivation, emotional state, and expectations in conjunction with long-term memory. As a result of this step selective attention is updated based on the new goal and the updated situation and goal are passed to the action selection module.

3. Long-term Memory

Long-term memory stores information learned over-time for future retrieval based on the situation. Reward histories from prior action selections as well as long-term beliefs and issue stances are maintained in long-term memory. Relevant actions for given situations as well as mappings of equivalent situations can be returned from long-term memory based on need in a given situation.

4. Action Selection

Situation based action selection facilitates the reduction of the state space of the model through the notion of equivalent states being categorized as unique situations. For each unique situation there exists a set of relevant candidate actions. Actions have an associated activation level provided by a utility based RL algorithm (Sutton & Barto, 1998). If the agent has enough experience, defined as a specified number of trials of each action, then the agent action selection is controlled by a softmax function, such as the Boltzman distribution, with a greedy setting, replicating recognition prime decision making (Klein, 1993). If the agent has some level of experience in the situation then action selection can still be conducted using the softmax function, but with an exploratory setting. If the agent has no experience in the situation, then mental simulation is conducted, with the agent using available knowledge regarding the environment to project future states based on the actions currently available (Klein, 1993; Kunde, 2005). An alternative to the case where sufficient experience is not present is to base the decision making mode on the risk level associated with the given situation. In this formulation, if the requisite experience to use recognition prime decision making is not present, when risk is low the agent simply uses the softmax function with an exploratory setting, while if risk is high the agent uses mental simulation.

B. GENERIC IMPLEMENTATION DESCRIPTION

This section provides an overview of the implementation of the conceptual model in discrete event simulation (DES) (A. H. Buss & Sanchez, 2002; A. Buss, 2002; A. H. Buss & Sanchez, 2005; A. Buss, 2009). The conceptual model was implemented by Mr. Harold Yamauichi of Roland's and Associates as part of TRAC-MTRY's research on Irregular Warfare. Discrete event simulation is a form of simulation that represents phenomena of interest from the real-world through with state variables, parameters, events and scheduling edges. It is distinguished from time-stepped models in its handling of time. In a time stepped model, time advances at a fixed-time increment throughout the course of the sim-

ulation. In a DES model time advance is controlled by a master event list, which advances time in uneven increments in accordance with the next scheduled event (A. Buss & Blais, 2007; A. Buss, 2001).

Event graphs provide a graphical representation of a discrete event simulation. In this form, nodes represent events and edges represent scheduling relationships between events. State variables change in a piecewise constant manner with the occurrence of events within the simulation in accordance with transition functions contained in events. The cognitive architecture consists of five main components at the top-level: PerceptUmpire, Perception, MetaCognition, ActionSelection, and LongTermMemory.

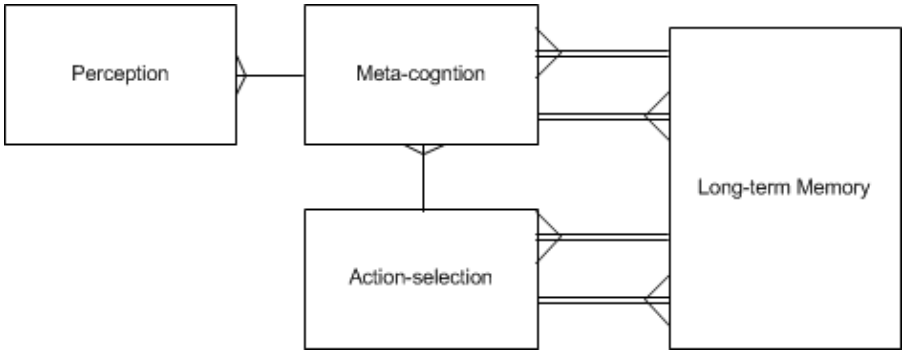


Figure 40: top-level view of DES cognitive architecture.

The PerceptUmpire serves as the interface between the agent and the environment. This class packages information about the environment and the internal state of the agent into atomic percepts that are provided to the agent in accordance with the rules provided by the modeler. This is the class that should require the most extensive work in regard to integrating the cognitive architecture into other environments. The PerceptUmpire not only provides a means of interfacing with the environment but also ensures that the modeler can control the information that the agent can access. This reduces the likelihood of the agent gaining access to unintended or unrealistic information from the environment. Since many of the capabilities being analyzed in support of the future force are intended to provide

timely information to decision makers, the need to ensure that only information that a particular command and control system would provide is included.

The PerceptUmpire consists of two events: ReceiveInformation and PerceptArrive. The ReceiveInformation event listens for events within the agent class and events within the environment, with supporting signatures for each event type that the agent has access to and packages the information into an atomic percept. This percept is then passed to the agent which treats it as an arrival event.

The Perception component filters percepts received based on relevancy, fills working memory and forms a situation object composed of the atomic percepts stored in working memory. The Perception component treats the arrival of each percept as an arrival event. The Perception component contains the following events: Arrival, RelevancyCheck, RelevantPercept, ProcesscurrentSituation, EndProcess, and StartMetaCognition. The Arrival event schedules a RelevancyCheck event. The relevancy check is intended to filter percepts based on their age and on their salience given the agents current goals. A percept is considered relevant if its time stamp is recent enough and its type is salient to the current goal. RelevantPercept is scheduled with a Boolean scheduling edge, and once the number of relevant percepts fills the available working memory a ProcessCurrentSituation event is scheduled. Working memory is constrained in accordance with the literature on human information processing. ProcessCurrentSituation creates a situation object and schedules EndProcess which in turn schedules a StartMetaCognition event. These events are further organized into three module: Selective attention, Working memory, and Situation formation.

In MetaCognition the situation object is used to determine current relevant motivations, assess the agents internal state in the given situation, select goals and methods to achieve them and updates long-term memory. From a high level meta-cognition contains two modules. MetaCognition listens for the StartMetaCognition event in the Perception component. StartMetaCognition takes the situation object formed in the Perception component and schedules an UpdateLongTermMemory event and an UpdateMotivation event.

The UpdateMotivation event takes the situation object and determines the change to the agents perceived motivations based on the new information. The view on motivation taken in this work is inspired by Kenricks work to update Maslows hierarchy of needs, see Figure 41,but the general framework need not be limited to this set for the general case (Kenrick, Griskevicius, Neuberg, & Schaller, 2010). An activation level for each motivation is calculated based on the agents perception of the needs associated with each motivation. As the needs for a given motivation are satisfied the activation level of that motivation is reduced. The UpdateMotivation event schedules a CognitiveAppraisal event. This event determines an overall level of satisfaction for the agent based on the current motivations. If the agent is completely satisfied all of its needs are met and its motivations are equally weighted. The CognitiveAppriasal event schedules a FormExpectation event. The FormExpectation event uses the current situation to determine the expected satisfaction level in the next most likely situation based on its experience. The difference between current and expected satisfaction is used to determine whether the agents outlook is optimistic or pessimistic. The CognitiveAppraisal event schedules a GoalsAndMethods event. This event determines the most relevant goal based on the motivations selects a top-level method to achieve that goal from the set of relevant methods using RL. If the goal has not changed from the current goal, then the agent continues with its currently scheduled behavior.

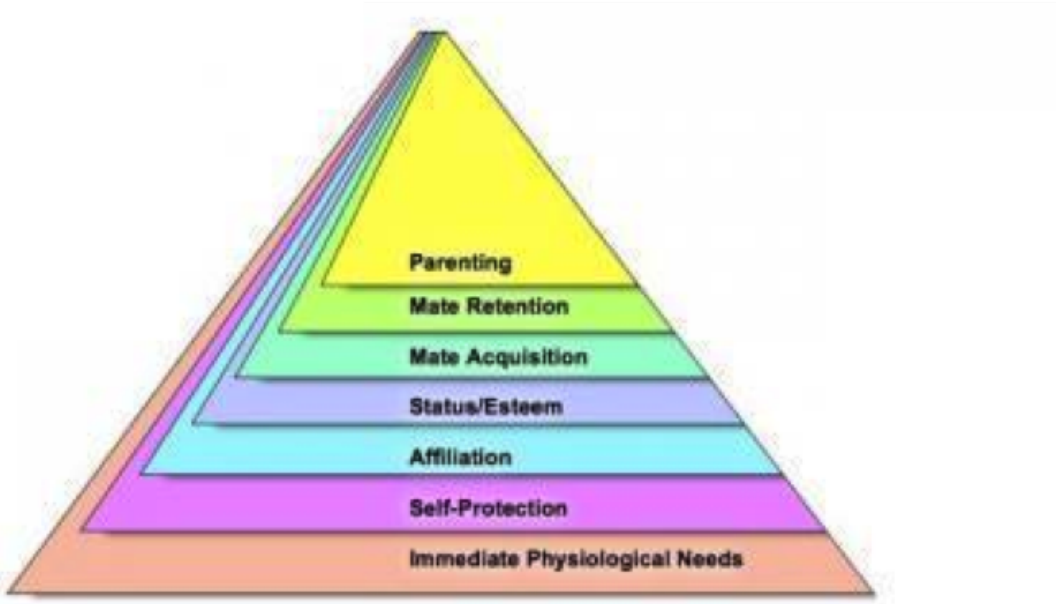


Figure 41: Kenrick's updated to Maslow's hierarchy of needs.

The LongTermMemory component contains the agents history based on its experience in the environment as well as long-term belief networks related to its view of the world. Event and reward histories are stored for use during RL. The belief networks document the agents world-view on certain issues and is represented in a Nave Bayes network. Otherwise the GoalsAndMethods event in turn schedules an IdentifyBehaviors event which returns the behaviors associated with the chosen method. The ActionSelection component identifies the behaviors associated with the chosen method and determines the appropriate method of decision making based on the agents experience with these behaviors. This component was inspired by Kleins recognition prime decision making. Once an action is selected the agent schedules the action within the environment. The ActionSelection component listens for an IdentifyBehavior event. The IdentifyBehavior event schedules an

IdentifyDecisionMethod event. Based on the experience of the agent in the situation, as reflected by the number of trials of each of the relevant actions associated with the selected method, either a RecognitionPrimedDecisionMaking event, an ExplorationLearning event, or a MentalSimulaiton event is scheduled. Each of these methods of decision making results in the scheduling of a SelectAction event which in turn schedules an InvokeAction event.

The cognitive architecture was implemented and tested by Harold Yamauichi of Roland's and Associates targeted for use in the Cultural Geography model. In the next section, we review the literature on social cognition in order to understand how cognitive architectures relate to the representation of group cognition, a relevant application areas where we have applied cognitive architectures to represent the decision making of a population segment within a conflict environment (J. Alt et al., 2009).

C. COGNITIVE ARCHITECTURES TO REPRESENT GROUP COGNITION

The concept of social cognition can take on varied meanings depending on the discipline of the researchers who are using it. In this research social cognition refers to the social processing and construction of information. The concept is grounded in the literature on psychology, with roots in the cognitive revolution, and is now largely accepted by contemporary social psychologists. The notion of social cognition also traces its roots to sociological theory, where as early as 1898, Durkheim proposed a theory of collective psychology, suggesting that members of social groups took part in a "collective conscience" and made use of a common knowledge base (Howard & Renfrow, 2006). The concept of prototypes, similar to Weber's notion of "ideal types", is supported by the use of a social information processing system, allowing group perceptions of other groups to influence their behavior in regard to the other group (Howard & Renfrow, 2006).

Social exchange theory, which assumes individuals and groups can be motivated by rewards, benefits from social cognition, allowing for the consideration of beliefs and attitudes in decision making (Cook & Rice, 2006; Howard & Renfrow, 2006). Note that this

also provides us a theoretical basis for the use of reinforcement learning at the group level. Expectation states theory attempts to explain the process by which humans and groups determine their roles and status in social situations, typically studied in task oriented groups (Correll & Ridgeway, 2006; Howard & Renfrow, 2006). In this context expectation states theory helps explain how expectations are formed regarding social identities within categorized groups based on status characteristics to eventually form representations of expected behavior for each group (Correll & Ridgeway, 2006). Rankings of power and prestige are generated based this "collective" reward system. Again we see a linkage to the use of rewards to shape behavior. Critical social psychology seeks to identify the role of power in social processes (Howard & Renfrow, 2006). The formation of a perception of other groups, central to the categorizations necessary to conduct the ranking needed for social interactions, leads to the need for social cognition. The elements of social cognition are broken into two categories by Howard et al., cognitive structures and cognitive processes (Howard & Renfrow, 2006).

In this summary we see that the notion of an information processing system present at the group level matches with the theory from social psychology and that the notion of behavior shaped by rewards exists at the group level as well.

1. Cognitive Structures

Cognitive structures serve as a store of information in verbal form, with belief, attitudes, and values as examples of early structural forms. Social cognition proposes that information regarding other individuals or groups is stored in a prototype, constructed through a synthesis of all information regarding the other group. These prototypes are used to rapidly assess members of these other groups. Exemplars provide an alternative to prototypes, but are based on specific experiences with the other group rather than the average. Schemas provide yet a third alternative, allowing the application of social knowledge to the entire information processing system through their impact on the organization of knowledge. Schemas can be used to store knowledge regarding individuals or groups, person schemas, about self (individual or group), self-schemas, regarding roles, and in relation

to events. Social representations have been proposed as a social alternative to schemas, being defined as knowledge that has been generated by consensus and shared with the group, "commonsense theories about how the world works (Howard & Renfrow, 2006)."

Memory studies have illustrated that memory is shaped by the social context and existing cognitive structures. Cohen in particular demonstrated this with experiments in which subjects were shown a video of a woman having dinner with her husband. Subjects were told the woman was either a waitress or a librarian and were asked to recall details regarding the video. The subjects responses were consistent with the role they perceived her to be filling, with memories of the actual content being replaced by the schema associated with that particular role. Wegner shows that memory occurs at the group level as well as the individual, with groups dividing information among group members (Howard & Renfrow, 2006).

The impact of language on social cognition is strong, since the mechanism for the development and storage of group memories is communication facilitated by language. Group perception of information transmitted in the same language depends on the situation and the categorization of the sending group (Howard & Renfrow, 2006). The social knowledge base for a group then might contain facts in natural language that carry specific meaning for that group distinguished from other groups. From a production system viewpoint, the processing of a fact by two groups might result in the activation of completely orthogonal rules by the production system based on the difference in the social knowledge base.

The exploration of ideologies, attitudes, beliefs, and behavior by social psychologists provides a starting point for the identification of social facts for a given population subgroup. Ideologies, values, and attitudes are distinguished in this literature by their level of abstraction. Attitudes are generally held toward objects, values tend to focus on notions such as freedom, and ideologies relate to sets of values and attitudes. Despite the difference in level of abstraction these concepts share the following attributes: 1) they are all evaluative expressed as a positive or negative value toward an object; 2) they are all subjective

based on the individual or groups perception; 3) they all can be either active or inactive based on the situation; 4) they are interrelated (Maio, Olson, Bernard, & Luke, 2006). Attitudes are more influenced through direct experience, while values and ideologies rely more on indirect experience with potential sources identified as the family, media, and cultural. Behavior has more closely been linked to attitudes than values or ideologies (Maio et al., 2006). Viewed from an information processing perspective then attitudes, ideologies, and beliefs maintained in social memory, might be used to interpret a given situation and select the appropriate action. Decisions regarding intergroup relations, shown to rely on social cognitive processes such as social categorization or stereotyping and the need for groups to view themselves positively, result from this collective information processing (Brubaker, Loveman, & Stamatov, 2004; Hogg, 2006). The central importance of ideologies, values, attitudes, and behavior in the development of group cognitive architectures captured in the following quote regarding the September 11, 2001 attacks:

> In particular, the extremists actions have been regarded as an inevitable consequence of their peculiar mix of Islam and conservative ideology, their lack of respect for innocent human beings, and their hatred toward the United States. In other words, the extremists behavior has been partly regarded as a product of their ideologies, values, and attitudes. (Maio et al., 2006)

2. Cognitive Processes

Cognitive processes in social cognition leverage knowledge organized in the cognitive structures (Howard & Renfrow, 2006). The social information processing system mirrors the standard information processing paradigm with subtle differences. Salience of a stimulus is known to be task dependent in visual information processing (Anderson, 2005). In the case of social information processing, the social meaning of percepts from the environment define the social salience of the information. In a similar manner, percepts regarding other groups that contradict expected behavior tend to be more socially salient (Howard & Renfrow, 2006).

The social cognition view of cognitive inference is geared for the use of the information gathered by an individual or group to make some social judgment (Howard & Renfrow, 2006). In a social context, we rely on prototypes of groups to make inferences about groups or group members we encounter. The group prototype we follow in regard to a given social entity also impacts the selective attention we allocate tof the individual or group (Howard & Renfrow, 2006). During a meta-cognitive step individuals or groups transform the perceived information using heuristic shortcuts as implied by the information processing model from cognitive science. Group meta-cognition has been empirically studied in the context of group problem solving, with the value of group meta-cognition increasing with the difficulty of the problem (Iiskala, Vauras, Lehtinen, & Salonen, 2011; Wilson, 2001).

Humans make decisions that tend to confirm their pre-existing belief, in spite of new evidence (Howard & Renfrow, 2006). Using limited information people tend to make causal inferences regarding the factors that produced a given outcome in a given situation, assigning characteristics to individuals or groups, associated with situations they encounter. Jones proposed that trait attributions resulted from the evaluation of observed situational behavior and prior expectations regarding other groups influenced by social norms. Several studies demonstrated that those viewed as being in positions of power had their behavior negatively categorized more frequently than others (Howard & Renfrow, 2006).

3. Sentiment

The impact of emotion on social information processing has been documented in a number of empirical studies (Jackson & Sullivan, 1989). Stets equates emotion and sentiment in her treatment of experiences that result from the combined influences of the biological, the cognitive, and the social and defines sentiment as "distinctly social in that individuals learn through socialization with others the names of the internal sensations they experience and the social norms regarding their appropriate expression (Stets, 2006). Stets, referencing Thoits, cites four related elements that comprise an emotion: 1) situational cues; 2) physiological changes; 3) visually expressive gestures; 4) the socially defined

label that describes the combination of the other three elements currently being experienced (Stets, 2006). Stets attributes a list of contemporary primary emotions to Turner: assertion-anger, aversion-fear, disappointment-sadness, and satisfaction-happy (Stets, 2006). These contemporary views are founded on work with at the individual, identifying five common emotions based on facial expressions, and work to map these primary emotions to the field of sociology. In the sociological view, combinations of these primary emotions can lead to secondary emotions, or social emotions.

Emotional intelligence and emotional competence both describe the role that emotion plays in information processing. Emotional intelligence is described as the ability to perceive and express emotion, to integrate emotion into reasoning, to understand emotion, and to control or manage emotion (Stets, 2006). The biological view of emotion focuses on emotions as expressions of internal feedback from the body, while the social structural approach relates sentiment to changes in power and status. The cultural perspective traces emotions to cultural norms, which provide information on how to feel in a given situation. A difference in generated emotions and culturally expected emotions describes emotional deviance in this paradigm. The symbolic interactionist framework relates emotion most closely to internal processes within individuals. Affect control theory, one variant of this paradigm, describes how emotions arise when internal feedback conflicts with the individuals perceived identity in the situation (Stets, 2006). This is not dissimilar to the cultural view.

4. Group Behavior and Deviance

Oakes suggested that group collective properties can explain behavior, supported by Allison and Messick who provide an account of the attribution of opinions derived from group decisions to individual members, even when collective decisions were not made (Howard & Renfrow, 2006). Power relationships also influence the attributions of group members and subsequent intergroup relationships. Attribution processes between groups are shaped by historical, economic, and political contexts of intergroup relations and is consistent with Fisher's narrative paradigm. This favoring of closely related groups results

in a greater allocation of selective attention to those groups categorized as most like the observing group, while relying on more basic schemas for those groups that are not as relevant. One result of this difference in the level of detail is illustrated by the extreme evaluations we tend to place on those groups with which we interact and attend to less frequently (Howard & Renfrow, 2006). Visual traits often are used for categorization, and when these traits occur in conjunction with a difference in resources the group with more resources will usually be assigned more status. This categorization process leads to the formation of group identities, which tend to be more relevant for those that are disadvantaged than high status groups, consistent with social identify theory (Howard & Renfrow, 2006). This is consistent with work on deviance, which uses the shared normative expectations of groups as mechanism by which groups evaluate behaviors and world views of individuals or other groups (Kaplan, 2006). Those that deviate from established norms are described as doing so from motivated deviance, usually due to a lack of motivation to adhere to social norms, or unmotivated deviance, usually in spite of the individuals efforts to adhere (Kaplan, 2006). Groups define social norms, which eventually might become legitimatized as laws of a country established by the dominant group. Motivations and goals play an important role in the adherence of members of a group to established norms.

The use of cognitive architectures to represent group cognition within social simulation has not been explored extensively in the literature, but the use of this type of information processing framework at the group level seems reasonable given the social psychology literature. The following section will describe the empirical performance of the cognitive architecture previously described when embedded within the CG model.

D. APPLICATION OF THE REINFORCEMENT LEARNING AND A PRACTICAL COGNITIVE ARCHITECTURE WITHIN THE CULTURAL GEOGRAPHY MODEL

This section describes the application of RL to the Cultural Geography model beginning in 2010 with the incorporation of RL methods into the representation of the Theory of Planned Behavior to drive action selection (S. Papadopoulos et al., 2013). Subsequently,

this research also developed an RL based representation of trust that was incorporated into the agent-level decision regarding the selection of targets for communications (Pollock et al., 2011). A methodology to develop scenarios from existing data was also provided and demonstrated using open source data sources (McKaughan, 2011). Finally, a proof of principle cognitive architecture based on the conceptual model described incorporating RL was developed and implemented (J. K. Alt et al., 2011).

1. Initial Application of Reinforcement Learning within CG

Initially, RL was used to control the behavior of threat agent, to more clearly understand how RL might be applied within a complex social simulation. The scenario used represents an area of Kandahar province in Afghanistan, with a civilian population of 350 agents, insurgent, host nation, and stabilizing forces, which communicate within a dynamic social network. The population agents take actions to meet basic needs. Threat forces attempt to reduce the satisfaction of the population on a set of issues related to stability, while host nation and stabilizing forces seek to improve the populations satisfaction on this issue set. Figure 48 shows the methodology used to develop the scenario and conduct analysis, while Figure 49 shows a functional decomposition of a generic population agent.

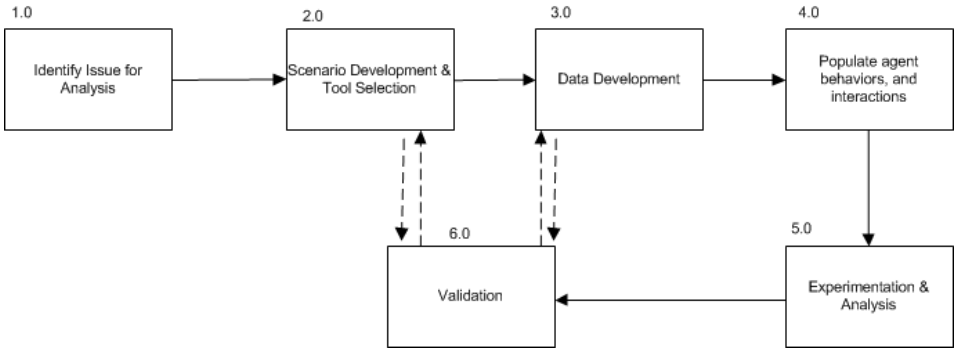


Figure 42: Analysis methodology for close formed use case.

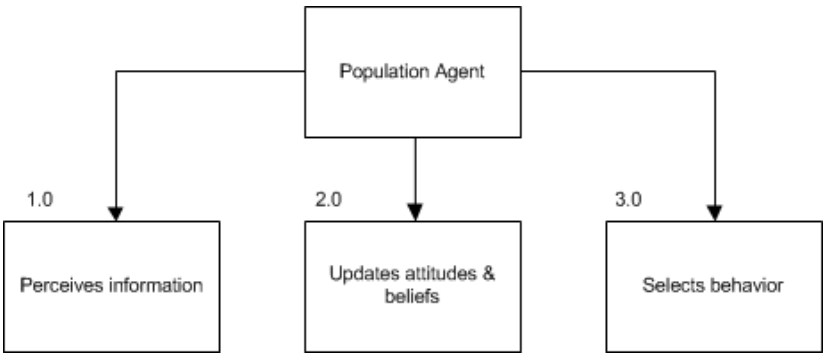


Figure 43: Population agent functional decomposition.

The initial implementation of RL within the CG model focused on enabling the threat agents to use RL to control their action selection. There is one insurgent agent per region each with four action choices:

- DoNothing: The agent performs no action.
- KillCivilServant: The agent makes an assassination attempt against a civil servant.
- IED: The agent plants an improvised explosive device (IED) against any target.
- $\mathrm{IED}_{\mathrm{ANSF}}$: The agent plants an IED targeting the Afghan National Security Forces (ANSF).

The problem is similar in nature to a noisy bandit problem with a reward function providing the insurgents feedback following each action opportunity in the presence of a variety of competing actions selected by the host nation and stabilizing forces. In this simple case two agents are compared: an RL agent, Tal1, and a standard agent, Tal2. The distribution of actions selected by each agent is shown in Figure **??**. KillCivilServant provides the best reward to the threat agent and we see that the threat agent does select this action more often than the standard agent.

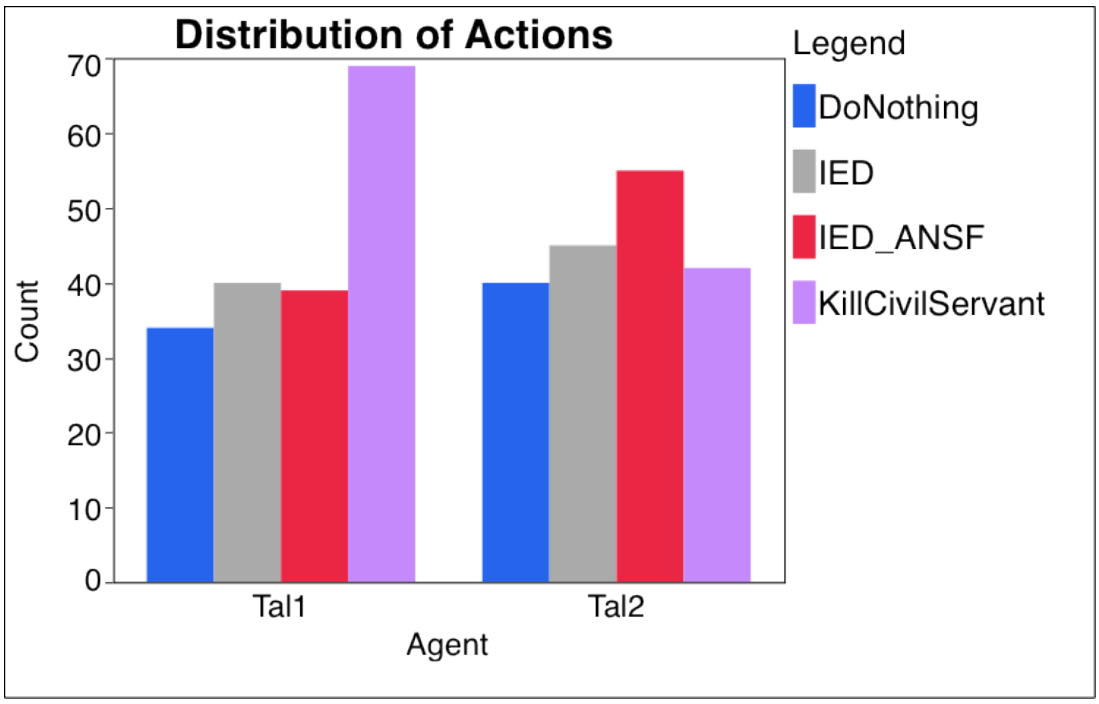


Figure 44: Distribution of threat actions.

This simple case illustrated that the RL agent responded to the dynamics of the environment rather than performing its actions according to a predetermined script. This approach served to reduce scenario and data development time and provide more realistic results. Building on the initial implementation of this technique in guiding the behaviors of insurgent actors, the next section describes the expansion of this research to explore the applicability of the use of reinforcement learning in the representation of the theory of planned behavior.

2. Representing Theory of Planned Behavior

Previous iterations of the CG model implemented the theory of planned behavior through the use of Bayesian belief networks. This approach, which resulted in a separate belief network for each agent, required extensive subject matter expert input and served to greatly increase runtime during execution. In order to reduce the data requirements and to

link the agent responses more closely with the dynamics of the model, an implementation of the theory of planned behavior that relied on RL was developed and incorporated into the model. This resulted in reduced runtime and a reduced data development process and model output that responded to the dynamics of the model. The research discussed here has been accepted for journal publication (S. Papadopoulos et al., 2013).

The Theory of Planned Behavior (TPB) provides an empirically grounded conceptual model of the formation of human behavioral intention (Ajzen, 1991). The theory of planned behavior states that behavioral intentions are formed by a combination of input from perceived subjective norms toward the behavior, internal attitude toward the behavior, and perceived behavioral control regarding the behavior. Subjective norm (SN) refers to the opinions of those in the individuals peer group regarding the behavior. The components SN are: the normative belief strength, nb, and motivation, m, to comply with the normative belief summed across the relevant members of the individuals social network, n.

$$SN = \sum_{i}^{n} N_i m_i \tag{42}$$

Attitude (A) refers to the individuals own beliefs regarding the behavior in question. The attitude is expressed as the product of belief, b, and the outcome evaluation, e, an evaluation of the value of the potential outcome (Ajzen, 1991).

$$A = \sum_{i}^{n} b_i e_i \tag{43}$$

Perceived behavioral control (PBC) describes the individuals perception of the likelihood that they will succeed in the execution of the behavior given that they attempt it. Control beliefs, cb, serve as the likelihood estimate, while perceived facilitation, pf, provides the value estimate (Ajzen, 1991).

$$PBC = \sum_{i}^{n} C_i p_i \tag{44}$$

The model, as formulated by the (Ajzen, 1991), is an expected value model, with a likelihood and payoff derived for each component. The linear sum of these components provides a behavioral intention (BI) score for that individual for the behavior in question.

$$BI = A + SN + PBC \tag{45}$$

In order to adapt the theory of planned behavior for use in the CG model we implement the theory of planned behavior using a reinforcement learning based approach. Perceived behavioral control (PBC) can be thought of as the discounted reward history for the actions associated with the behavior of interest.

$$PBC_a = \sum_i \lambda^{t_i - t} u_i H(t_i > t), \forall a \in A \tag{46}$$

The subjective norm (SN) is represented as the average activation levels of the k nearest neighbors within the social network.

$$SN_a = \frac{\sum_{i=1}^{k} PBC_{ai}}{k}, \forall a \in A \tag{47}$$

The attitude (A) toward the behavior is likely the most difficult to generalize, since social simulations represent this in a variety of ways if at all. For the purposes of this example, it can be thought of as the utility associated with the behavior, B, at time t, the current simulation time.

$$A_a = U_{B_t}, \forall a \in A \tag{48}$$

In the CG model, the attitude toward a given behavior is drawn from a set of belief and issue stances or strengths that are updated dynamically over the course of a simulation run. The formulation shown below then is equivalent to a multi-attribute utility function that considers the agents internal state, A, the opinion of others, SN, and the external reward or success history, PBC, associated with a behavioral action selection.

$$\beta_a = w_1 A_a + w_2 SN_a + w_3 PBC_a, \forall a \in A \tag{49}$$

$$\text{s.t.} \sum_{i=1}^{i=3} w_i = 1.0 \tag{50}$$

This formulation was used in support of current operations analysis of the Kandahar province of Afghanistan and is the subject of an upcoming journal article (S. Papadopoulos et al., 2013). Results of simple case study analysis verify that the RL code behaves as predicted. In this case, the case study involved a small experiment that controlled the level of water at each of the servers in a provinces. As the level of water was reduced in one area, rather than simply continue to execute the same action despite a negative outcome, as in previous versions of the model, the agents adapt their behavior and try alternatives, eventually identifying which servers still have water. While this may sound simple, the use of agents that actually respond to the dynamics of the environment greatly increased the face validity of the model output. The use of RL also reduced the scenario construction time and data requirements while reducing runtime over previous model versions.

3. Incorporation of a Cognitive Architecture

The cognitive architecture, implemented by Rolands and Associates, was incorporated into the CG model and used in support of the FY11 Irregular Warfare Tactical Wargame (IW TWG). During the war-game the CG model is used in a human in the loop fashion, with human players making action selections which become inputs to the modeling suite. The simulation runs for a weeks worth of game time and results from a single run are provided back to the player. The action space for the population agents in this use case is very small and the state space in this implementation was ill-defined. The code and the cognitive architecture functioned as expected, controlling the level of information provided to each agent through the percept mediator and allowing the agents to form a notion of state and map states to actions. Agent's that use the cognitive architecture would also be well-suited for use as autonomous red or blue forces in a close loop setting or as

semi-autonomous red or blue forces in a war game. Should the population agents be provided a richer action space, they might warrant the use of more sophisticated autonomous behavior. Note that the scenario configuration and definition of the reward signal must be accomplished for each application of the CG model. The implementation served as the centerpiece of the FY11 IW TWG facilitating the player stimulus regarding population output which drove the conduct of the war-game. This effort was recognized with the Army Modeling and Simulation Office Team Award in 2011.

Rigorous testing of the integrated code is ongoing at TRAC-MTRY and initial results verified that the code is functioning properly and that the scenario file was not properly configured in a number of respects in the previous iteration. The use of cognitive architectures to represent goal driven behavior within simulation and the incorporation of reinforcement learning to prevent overly brittle solutions continues to be an active area of research.

E. CONCLUSIONS

In conclusion this chapter documented the development of a practical cognitive architecture and the use of RL in the representation of human population agents within an agent based social simulation. RL was first used to make decisions for threat forces in a closed loop setting and later used to represent the theory of planned behavior to drive decision making for population agents in a close looped setting. These innovations were used to support an Irregular Warfare Tactical Wargame (IW TWG) in 20120 and an analysis of the population along the Pakistan-Afghanistan border. Finally, a version of the cognitive architecture was incorporated into the CG model and used in support of IW TWG 2011, which received a Army Modeling and Simulation award for excellence in analysis. Scenario configuration proved to be challenging primarily due to scenario initialization files and limited data availability and the use case. While CG was developed for use in close form its use case migrated to a human in the loop war-game, but the design focus remained on the close form case rather than the stimulation of human players within a game-based

setting. While the models, including the cognitive architecture, functioned as intended, the signal provided to the players was noisy due to lack of calibration. This issue was identified in after action reviews and strategies to correct his have been implemented by the team along with a refocus of model development to a human in the loop use case.

The use of RL techniques in this area in conjunction with a practical cognitive architecture or other construct to maintain domain knowledge provides a conceptual model for human behavior that has the potential to be validated and that represents the essential elements of human information processing.

VI. CONCLUSION

This chapter provides a summary of the contributions of this dissertation and directions for future research in this area.

A. SUMMARY OF CONTRIBUTIONS

This research provided insights into the application of model-free reinforcement learning algorithms within Department of Defense simulation models. The application of these algorithms to several use cases was described:

- Human behavior representation.
- Assignment and scheduling of resources.
- Validation of reward structures within training simulations.

The research developed a novel reinforcement learning algorithm, Direct-Q Computation, designed to speed reinforcement learning by directly addressing the temporal credit assignment problem in reinforcement learning. This algorithm leverages properties of maximum likelihood estimation to develop an estimator that minimize bias resulting in faster learning in sequential decision making tasks with delayed reward when compared with other model-free approaches.

1. Direct-Q Computation

The primary contribution of this research is the novel use of the exponentially weighted average reward as an action-value estimator in reinforcement learning systems in order to address the temporal credit assignment problem in reinforcement learning. This relatively straightforward approach improves learning speed over dominant existing approaches in task environments with noisy and delayed reward signals and improves performance in non-stationary environments, while reducing the number of parameters required

to be specified by the system designer from 3 to 2. Many real-world applications fall into this category of problem and in these cases delays in learning or recovery can result in control system failures and lost resources. The results provide the modeling and simulation community with a method that speeds learning in these challenging cases, reducing the time required to train autonomous software agents and the time required for agents to adjust to changes in their environment. These performance results carry over into each of the multiple modeling and simulation application areas examined in this research.

2. Enabling Adaptive Behavior in a Combat Simulation

A fourth major contribution of this research a methodology and application of reinforcement learning to represent human decision-making within a combat simulation. This straight-forward approach provides an empirically developed conceptual model of human decision making, important for eventual model validation, that facilitates dynamic decision making and allows agents to learn from interaction with their environment. This approach incorporates the novel use of reinforcement learning within hierarchical task networks, providing the potential to enable adaptive decision making within complex behaviors. This has particular relevance for enabling agents that adapt to the behavior of an opposing force, as human decision makers do, as opposed to relatively brittle scripted methods currently in use.

3. Maximizing the Value of a UAV Schedule from a DES

RL algorithms provide an alternative approach to dynamic programming techniques currently employed for scheduling and assignment of resources in TRAC's assignment and scheduling tool. This simulation is used to produce a feasible schedule for unmanned aerial assets that maximizes a value function by correctly pairing platforms with mission demands in the context of a combat scenario. The current approach employs a linear program that maximizes value over a finite-time horizon, but fails to provide a feasible schedule that maximizes value in cases where high value targets appear beyond the time horizon or where high value emergent targets become available following the initial allocation. Fur-

ther, this approach fails to learn to recognize the cues leading to these situations, as a human decision maker would over-time, and makes these mistakes consistently resulting in feasible, but non-value maximizing schedules. The value of the feasible schedule produced in different combat simulations for a given mix of unmanned platforms is used to inform acquisition decisions regarding unmanned assets, so the current tools limitations directly impact the representation of the value of a given mix to senior decision makers. Since the difference in the value lost to these cases varies across mixes, the analyst cannot know how this systematic issue effects results in the aggregate.This research demonstrates the use of reinforcement learning to address these cases and an approach that relaxes the strict requirement for fully observable demands currently imposed on the simulation.

4. Verification of Reward Structure in Training Simulation

A major contribution of this research is a novel methodology and example application of the use of reinforcement learning as a means of verifying the reward structure of a training simulation. The reward structure in a training simulation directly impacts trainee learning-time and outcomes. A weak reward signal will result in slower learning and a reward signal that rewards trainee actions that are not consistent with learning objectives will result in learning the trainee learning the wrong objectives. This research demonstrates the use of reinforcement learning to examine the reward structure and produce an example of the learned behavior, or policy, that can provide the training simulation designer feedback on the student behaviors rewarded by the training simulation prior to the simulation ever touching student hands, allowing the developer the opportunity to identify and correct deficiencies prior to fielding. This research contributes a repeatable methodology for the use of RL in this use case.

5. Practical Cognitive Architecture

RL algorithms used in conjunction with cognitive architectures provide a traceable means enabling autonomous behavior while representing human decision making in DoD simulation models. The use of these methods within social simulations, as suggested by the

NRC and others, provided increased transparency when implemented within the CG model and significantly reduced data development and improved run-time over previous methods, while producing similar results. A fifth contribution of this research is the development and application of a novel practical cognitive architecture that facilitates the representation of human information processing and the inclusion of domain knowledge in a structured manner that enables the selective use of goal-driven reinforcement learning to represent human decision making. The cognitive architecture provides an understandable framework to incorporate the effects of perception, working memory, and dynamic goal-setting within simulation agents. This is particularly relevant for analysis topics related to the value of information or the impact of networked sensors. The cognitive architecture also has relevance to the representation of civilian behavior in conflict areas, where the analysis focuses on the beliefs and interests of a population and the cognitive architecture provides a organizing construct. This contribution was incorporated into a social simulation used to facilitate war-games that received a 2011 Army Modeling and Simulation Office award for excellence in analysis.

B. FUTURE RESEARCH

Future research could extend the results of this dissertation by incorporating techniques to dynamically adjust the ratio between exploration and exploitation and by integrating the cognitive architecture within other simulation models, such as COMBATXXI. Balancing exploration and exploitation is critical to the performance of reinforcement learning systems and is a problem encountered across multiple disciplines. Future work in this area is vitally important to both improving agent performance and to understanding how humans accomplish this challenging task.

Perception of state is essential to decision-making and learning, but the state representation must remain as sparse as possible to speed learning in reinforcement learning systems. Developing methodologies to identify the key components of the environmental state in different decision situations is key to understanding how decision-makers reason in

uncertain environments and the information required to represent at the agent-level in order to facilitate more realistic representations of human behavior in DoD simulation models.

The application of reinforcement learning methods to facilitate sequential design of experiments is an interesting application area not explored in this research, but of interest to the modeling and simulation community. In this research we discuss the application of reinforcement learning to one aspect of training simulation, but a separate application area could involve the use of reinforcement learning agents to drive the pace of instruction based on feedback from the trainee, serving as intelligent tutors.

In this research we have sought to demonstrate the broad applicability of reinforcement learning within DoD models and simulations, but there is still much work to be done and to realize the full potential of these simple yet powerful techniques within defense modeling and simulation applications.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A: FIRST-VISIT AND EVERY-VISIT MONTE-CARLO

Algorithm 11 First-visit MC (Szepesvari, 2010)

FirstVisitMC(τ, V, n)

Parameters: $\tau = (s_0, r_1, ..., r_T, s_T)$ is a trajectory with s_T being an absorbing state, n is the number of times the long-term value estimate, V, was previously updated, γ is the discount rate, α is the learning rate, and t is the current trial.

sum$\leftarrow$0

for $t = 0$ to $T - 1$ **do**

    $sum \leftarrow sum + \gamma^t r_{t+1}$

end for

$V \leftarrow V + \frac{1}{n}(sum - V)$

return V

Algorithm 12 Every-visit MC (Szepesvari, 2010)

EveryVisitMC($s_0, r_1, s_1, r_2, ..., s_{T-1}, r_T, V$)

Parameters: s_t is the state at turn t, r_{t+1} is the reward associated with the t^{th} transition, T is the length of the episode, V is the array storing the current value function estimate, γ is the discount rate, α is the learning rate.

sum$\leftarrow$ 0

for $t \leftarrow T - 1$ down to 0 **do**

    sum$\leftarrow$ $tr_{t+1} + \gamma$ sum

    $\delta(s_t) \leftarrow$ sum

    $V(s_t) \leftarrow V(s_t) + \alpha(\delta(s_t) - V(S_t))$

end for

return V

Algorithm 13 TD(λ) (Szepesvari, 2010; Sutton & Barto, 1998)

1: TD(λ)

2: Parameters: s, the current state; V(s), current value estimate of current state; e(s), current eligibility to receive credit of current state; γ, discount rate; α, learning rate; λ, decay rate.

3: Initialize π (ie. $\epsilon - \mathtt{greedy}, \mathtt{Boltzmann}$).

4: Initialize $V(s)$ arbitrarily, $e(s) = 0$ for all (s).

5: Return a using $\pi(s)$.

6: Take action a, observe reward r, and next state, s'.

7: $\delta \leftarrow r + \gamma V(s') - V(s)$.

8: $e(s) \leftarrow e(s) + 1$.

9: **for** For all s: **do**

10: $\quad V(s) \leftarrow V(s) + \alpha\delta e(s)$

11: $\quad e(s) \leftarrow \gamma\lambda e(s)$

12: **end for**

13: $s \leftarrow s'$

14: Increment t. If $t \leqslant T$, go to line 3.

APPENDIX B: COMPARISON OF ϵ-GREEDY AND BOLTZMANN EXPLORATION

It is useful to provide a more detailed explanation ϵ-greedy and Boltzmann exploration policies in order understand the true nature of their impact on RL performance when coupled with an action-value estimator. The choice of exploration policy is a significant factor in the performance of the overall learning system by all metrics, which is intuitive since the exploration policy essentially controls the sampling strategy of the learning system. Assume two independent learning agents are operating in an MDP as defined previously. For simplicity let it be a single-state MPD, $S = \langle s_0 \rangle$, with a finite action set consisting of only two actions, $A = \langle a_0, a_1 \rangle$, and that both agents are using the same action-value estimation algorithm and are provided with exactly the same estimates, $Q(s, a)$. Assume one agent employs an epsilon-greedy policy, with fixed ϵ , and the other agent employs a Boltzmann exploration policy, with fixed τ. Let the true value of each action be q_0 and q_1 respectively and let the true value plus estimation error be,

$$\begin{aligned} Q(s_0, a_0) &= q_0 + e_0, \\ Q(s_0, a_1) &= q_1 + e_1 \end{aligned} \tag{51}$$

, for each action choice with $q_0 > q_1$. To simplify notation for this single state case we will refer to $Q(s_0, a_0)$ and $Q(s_0, a_1)$ as Q_0 and Q_1.

Case 1. Let the estimation error associated with each estimate be equal, $e_0 = e_1$ and of the same sign. In this case since the estimation errors are equal we see that $Q_0 > Q_1$. Then we see that epsilon-greedy will always choose Q_0 with probability ϵ and Q_1 with probability $1 - \epsilon$. So the expected number of exploratory actions is ϵT , where T is the number of trials in the episode. In the case of the Boltzmann exploration policy the

probability of selecting each action is given by,

$$\begin{aligned} P(\pi(s_0) = a_0) &= \frac{\exp \frac{Q_0}{\tau}}{\sum_a \exp \frac{Q_a}{\tau}} \\ &= \frac{\exp \frac{q_0+e_0}{\tau}}{\sum_a \exp \frac{q_a+e_a}{\tau}} \\ P(\pi(s_0) = a_1) &= \frac{\exp \frac{Q_1}{\tau}}{\sum_a \exp \frac{Q_a}{\tau}} \\ &= \frac{\exp \frac{q_1+e_1}{\tau}}{\sum_a \exp \frac{q_a+e_a}{\tau}} \end{aligned} \tag{52}$$

Since we know that $q_0 > q_1$ and that $e_0 = e_1$ we see that, $\frac{q_0+e_0}{\tau} > \frac{q_1+e_1}{\tau}$ for fixed τ , but this does not imply that $P(\pi(s_0) = a_0) > P\pi(s_1) = a_1)$ since we have not specified enough information regarding τ. Dropping the error terms since they are equal, we know that the probability of selecting either action will be equal when,

$$\exp \frac{q_0}{\tau} = \exp \frac{q_1}{\tau}$$

$$\frac{q_0}{\tau} = \frac{q_1}{\tau} \tag{53}$$

$$\frac{q_0 - q_1}{\tau} = 0 \tag{54}$$

So we see that since the numerator in this case is a positive constant, since $q_0 > q_1$, that the only way to satisfy the condition for equal probabilities is as the denominator goes toward infinity.

$$\lim_{\tau \to \infty} \frac{q_0 - q_1}{\tau} \to 0 \tag{55}$$

The size of τ required for this condition to hold for our case depends on the magnitude of the interval $q_0 - q_1$. Recall that for the exponential function,

$$\lim_{x \to 0} \exp x \to 1 \tag{56}$$

and that for our case we see that for sufficiently large that,

$$P(\pi(s_0) = a_0)\ P(\pi(s_0) = a_1) \tag{57}$$

and the probability of taking an exploratory action in general is,

$$P(\pi(s_0) = a_1) = \frac{\frac{\exp \frac{q_1}{\tau}}{\sum_a \exp \frac{q_a}{\tau}}}{\exp \frac{q_0}{\tau} + \exp \frac{q_1}{\tau}} \tag{58}$$

and that the probability of taking an exploratory action with the Boltzmann policy and the epsilon-greedy policy will be the same when,

$$\epsilon = \frac{\exp \frac{q_1}{\tau}}{\exp \frac{q_0}{\tau} + \exp \frac{q_1}{\tau}} \tag{59}$$

which we can restate as,

$$\epsilon = \frac{1}{1 + \exp \frac{q_0}{\tau} - \frac{q_1}{\tau}} \tag{60}$$

let $q_0 - q_1 = \delta q$, and we can say,

$$\begin{aligned} \epsilon &= \frac{1}{1 + \exp \frac{\delta q}{\tau}} \\ \epsilon\left(1 + \exp \frac{\delta q}{\tau}\right) &= 1 \\ \epsilon \exp \frac{\delta q}{\tau} &= 1 - \epsilon \\ \tau &= \frac{\delta q}{\ln(1-\epsilon) - \ln(\epsilon)} \end{aligned} \tag{61}$$

This defines the temperature of the Boltzmann strategy in terms of epsilon and the difference between the action values for the case of equal errors in the estimate, both in the same direction. Note that this definition fails for the case where $1 - \epsilon = \epsilon$, and that $\epsilon \in (0, 1)$.

As $\epsilon \to 1$, the denominator goes toward negative infinity and as $\epsilon \to 0$, the denominator grows toward positive infinity, and in fact the function is symmetric about the fixed point at $\epsilon = 0.5$. With fixed numerator this yields a symmetric function for the temperature, producing positive temperature parameter values for low exploration probabilities and negative values when the probability of exploring is greater than 0.5. Setting the temperature parameter using this formulation would ensure that the probability of taking an exploratory action using a Boltzmann policy was equal to the probability of taking an exploratory action using the epsilon greedy policy and the expected number of exploratory actions using a Boltzmann strategy with its temperature parameter set in accordance with the formula above would equal ϵT making it equivalent to the epsilon-greedy case. The insight this provides us is that the difference between the action-value estimates is important for understanding the behavior of the Boltzmann policy as it is typically employed and the importance to the Boltzmann policy of being coupled with value estimation algorithms that minimize error or at least treat it in a systematic manner. We see from,

$$\epsilon = \frac{1}{1 + \exp \frac{\delta q}{\tau}} \tag{62}$$

that as $\delta q \to \infty$ that the probability of choosing the exploratory action goes toward zero given a fixed temperature parameter. This demonstrates the importance of the accuracy of the value estimator in the behavior of Boltzmann exploration and we see that when the difference between the value of the best action and the next best action are small that the probability of choosing an exploratory action will be relatively higher.

Case 2. Let the estimation errors be unequal such that, $e_0 < e_1$. Recall that $q_0 > q_1$ shows the relationship between the true values of the actions for our single state

MDP and that the value of the state action pair as perceived by the agent is,

$$
\begin{aligned}
Q(s_0, a_0) &= q_0 + e_0 \\
Q(s_0, a_1) &= q_1 + e_1 \qquad (63) \\
& \qquad (64)
\end{aligned}
$$

We begin again with the epsilon-greedy case, which will choose the action with the maximum perceived value with probability $1 - \epsilon$. In order for the epsilon approach to choose the correct action, the values must satisfy,

$$
\begin{aligned}
q_0 + e_0 &> q_1 + e_1 \\
q_0 - q_1 &> e_1 - e_0 \qquad (65) \\
\frac{q_0 - q_1}{e_1 - e_0} &> 1 \qquad (66) \\
\frac{\delta q}{\delta e} &> 1 \qquad (67)
\end{aligned}
$$

,so that the difference between the true values are greater than the difference between the errors associated with the estimates. The effect on the Boltzmann strategy is seen on the probability of choosing the exploratory action below,

$$
\begin{aligned}
P(\pi(s_0) = a_1) &= \frac{\exp \frac{Q_1}{\tau}}{\sum_a \exp \frac{Q_a}{\tau}} \\
&= \frac{\exp \frac{q_1 + e_1}{\tau}}{\sum_a \exp \frac{q_a + e_a}{\tau}}
\end{aligned} \qquad (68)
$$

and from our previous case we saw that this could be simplified to,

$$
\epsilon = \frac{1}{1 + \exp \frac{q_0}{\tau} - \frac{q_1}{\tau}} \qquad (69)
$$

and that,

$$\delta 1 = (q_0 - q_1) - +(e_0 - e_1) \tag{70}$$

Since $e_0 < e_1$ and $q_0 > q_1$ for this case, we see that the error term is negative and serves to reduce the magnitude of δq , which has the effect of increasing the likelihood of exploration since the difference between the two estimates is reduced for fixed temperature. Depending on the size of the error this could cause the exploration policy to take more exploratory steps, despite an accurate estimate of the better action, to improve its estimate, reducing overall system performance. We see then that any error in action-value estimation that serves to contract δq will induce exploratory behavior, while error that serves to increase this interval will induce more greedy behavior. Note that this case is often encountered in practice since it is common that the best action will be sampled more often resulting in a lower approximation error than actions sampled less often.

APPENDIX C: BAYESIAN OPTIMAL POLICY FOR N-ARM BANDIT

The total regret and regret per turn have been used to gage the performance of learning algorithms in the bandit problem previously. They make use of the expected utility of choosing the best alternative at each turn in comparison to the expected utility of the chosen path. As a means of gaining further insight into the specific problem of when to explore we expand the state tree for the 2-arm bandit problem and solve for the sequence of choices that lead to the best solution out to depth 10. The result is a set of nodes whose state is a tuple consisting of $\langle n_a, k_a, n_b, k_b \rangle$, where n is the number of times that an arm, a or b, has been chosen, and k is the number of times the arm has paid off. The first node in the tree then is the tuple $\langle 0, 0, 0, 0 \rangle$. Children are added to the parent nodes reflecting all states that are possible to reach from that node branching by a factor of four at each node like,

$$\langle n_a+1, k_a, n_b, k_b \rangle, \langle n_a+1, k_a+1, n_b, k_b \rangle, \langle n_a, k_a, n_b+1, k_b \rangle, \langle n_a, k_a, n_b+1, k_b+1 \rangle \tag{71}$$

The probability of visiting a state from the current state is calculated using the probability of success of each arm and a particular set of hyper-parameters, (α, β, w), set on initialization. The hyper-parameters are used to determine the probability of visiting any node, so the probability of transitioning from the root node to the child state $n_a = 1, k_a = 0, n_b = 0, k_b = 0$ is the probability of choosing A, w initially, time $(1-\alpha)$, the probability of not receiving a payoff after choosing A. The expected utility of each node is determined by either summing up the k hits for the parent node or determining the expected utility of the child nodes with the maximum expected utility of the child becoming the expected utility of the parent. The result is a tree that is completely expanded out to the specified depth with the expected utilities then rolled back up to the root node and the optimal choices of lever

identified at each depth. The resulting policy is harvested with an additional parameter added to the tuple reflecting the optimal choice at each state, $\langle n_a, k_a, n_b, k_b, c \rangle$.

Using this state information we can determine the points on the optimal path at which an exploratory action is taken from the perspective of an algorithm that does not know the underlying distributions of the arms, but is only relying on the observations contained in the original form of the tuple in order to learn the hyper-parameters. An exploratory action is defined as one in which the optimal choice, c, from a state transitions to the arm with the lower probability of success, $\min(p_a, p_b)$, where,

$$p_a = \frac{k_a}{n_a} \text{and} p_b = \frac{k_b}{n_b} \tag{72}$$

,$\forall n_a, n_b > 0$ and 0 otherwise.This approach provides the full set of nodes that belong to the optimal policy, $o \in P$, and the set of nodes that belong to the exploratory set, $e \in E$, that are also members of the set of nodes in the optimal policy for a given λ, α, β out to the specified depth.

Using this empirical data it is possible to empirically determine a policy, $f(n_a, k_a, n_b, k_b)$, to determine when it is optimal to explore. In order for this policy to be general and useful when the underlying distribution is not known requires that the policy be based on empirical data drawn from multiple combinations of λ, α, β . The added benefit of this approach, and the original motivation for the work, is that a tight bound on the expected discounted utility is produced that can then be used for comparison to empirical results using combinations of action-value functions and exploration strategies. Initial results of this approach applied to four combinations of α and β, $(\frac{31}{64}, \frac{33}{64})$, $(\frac{1}{8}, \frac{7}{8})$, $(\frac{15}{64}, \frac{17}{64})$, $(\frac{1}{8}, \frac{3}{8})$, across a range of λ, $\{0.1, 0.2, ..., 1.0\}$. This resulted in a set P consisting of 818 nodes and a set E consisting of 128 nodes, with w=0.5. No differences in optimal policy membership were observed as a result of variations in λ for the same distribution. Differences in optimal policy membership were observed between combinations of α and β, with the differences based on the magnitude of $\alpha - \beta$. The optimal choice of arm for all states when priors are known is obtained by updating the prior probability based on current observations.

$$P(\alpha, \beta|k_a, k_b)P(k_a, k_b) = P(k_a, k_b|, \beta)P(\alpha, \beta) \tag{73}$$

$$P(\alpha, \beta|k_a, k_b) \propto P(k_a, k_b|, \beta)P(\alpha, \beta) \tag{74}$$

and if $P(\alpha, \beta) = P(\beta, \alpha)$, then,

$$P(\alpha, \beta|k_a, k_b) \propto P(k_a, k_b|, \beta) \tag{75}$$

and,

$$P(\alpha, \beta) \propto \binom{n_a}{k_a} \alpha^{k_a}(1-\alpha)^{n_a-k_a} \binom{n_b}{k_b} \beta^{k_b}(a-\beta)^{n_b-k_b} \tag{76}$$

$$P(\beta, \alpha) \propto \binom{n_a}{k_a} \beta^{k_a}(1-\beta)^{n_a-k_a} \binom{n_b}{k_b} \alpha^{k_b}(a-\alpha)^{n_b-k_b} \tag{77}$$

If, for arms with equal rewards, we assume that one arm is always preferable, $\beta - \alpha > 0$, then the problem is to determine the perceived location of the best arm, β. So when,

$$P(\alpha, \beta|k_a, k_b) < P(\beta, \alpha|k_a, k_b), \mathrm{choose}A \tag{78}$$

$$P(\alpha, \beta|k_a, k_b) > P(\beta, \alpha|k_a, k_b), \mathrm{choose}B \tag{79}$$

$$P(\alpha, \beta|k_a, k_b) = P(\beta, \alpha|k_a, k_b), \mathrm{choose}A \tag{80}$$

Using these rules we correctly select the choice made at each of the nodes in our empirical set P. Using this information we can compare the decisions made by reinforcement learning algorithms to the optimal policy. This result provides us with a means of identifying the best arm out of the k arms when the priors are known. When the priors are unknown we need only know that there exists an arm, k^*, such that its probability of success, α_k^*, is greater than the probability of success of all other arms, $\alpha_k^* > \alpha_k$. The probability of the best arm being located in each of the k positions and the choice is the kth location with the highest probability, with arbitrary tie breaking. An examination of the member nodes of E results in observations useful to characterize the optimal time to

explore, remembering that the decision to explore is the result of an update that resulted in a false belief in the location of the optimal arm.

APPENDIX D: ANALYSIS OF DQ-C AND TD(λ)

Relating DQ-C to TD(λ)

TD(λ) serves as a foundational method bridging the gap between MC and TD techniques through the use of $e(s)$ to estimate $V(s)$ (Sutton & Barto, 1998). MC methods use the full sample return as the estimate,

$$R_{s_t}^{MC} = \sum_{i=0}^{L-1} \gamma^i r_{t+i}, \tag{81}$$

where L is the number of state transitions in after time t, while TD methods estimate the return by using the previous estimate,

$$R_{s_t}^{TD} = r_t + V(s_{t+1}), \tag{82}$$

where $R_{s_t}^{TD}$ is the estimate from s_t and r_t is the reward received going from s_t to s_{t+1}. These two estimates are examples of the more generalized n-step return, where for $n \geqslant 1$,

$$\begin{aligned} R_{s_t}^{(n)} &= r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \\ &\ldots + \gamma^{n-1} r_{t+n-1} + \gamma^n V(s_{t+n}), \end{aligned} \tag{83}$$

which uses the observations of n transitions and the previous estimate provided by the value function. TD(λ) uses all n-step returns simultaneously by incorporating a new parameter, $\lambda \in [0, 1]$ to provide the λ-return,

$$R_{s_t}^{\lambda} = (1-\lambda) \sum_{n=0}^{\infty} \lambda^n R_{s_t}^{(n+1)}. \tag{84}$$

The λ-return algorithm uses the λ-return to determine the increment to $V(s_t)$ for each step,

$$V(s_t) \leftarrow V(s_t) + \alpha[R_t^\lambda - V_t(s_t)], \tag{85}$$

with the update only occurring to $s = s_t$.

Pausing here we can see that $U_j(s, a)$ from DQ-C most closely compares to $R_{s_t}^{MC}$, differing in the manner in which point rewards are accrued, with DQ-C crediting each (s, a) pair with all future point rewards, indexing the sum on the number of point rewards, k, while $R_{s_t}^{MC}$ credits only s and indexes on the number of state transitions in the episode.The two approaches also differ in the manner in which γ is decayed, using the time between the visit to a given (s, a) in DQ-C versus the transition index in TD(λ).

$$U_j(s, a) = \sum_{i=1}^{k} r_i \gamma^{t_i - t_j} H(t_i < t_j),$$
$$R_{s_t}^{MC} = \sum_{i=0}^{L-1} \gamma^i r_{t+i}, \tag{86}$$

$U_j(s, a)$ clearly contrasts with $R_{s_t}^{TD}$, which bootstraps off its previous estimate. Since this forms the basis of $R_{s_t}^{(n+1)}$ which lies at the heart of the λ-return we see that DQ-C is distinctly different than the "forward view" of TD(λ). The manner in which $Q(s, a)$ is calculated in DQ-C differs from the λ-return as well, with the index of the summation in DQ-C being tied to visits to the (s, a) rather than steps forward in time from a given state.

The "backward view" of TD(λ) is typically used in implementation and increments $e(s)$ for each visit to a (s), see Algorithm 14. By contrast, DQ-C maintains an estimate for the value of each visit to a (s, a), directly assigning credit to each visit as point rewards are obtained in order to update the expected utility obtained from each visit.

Algorithm 14 TD(λ)

1: Initialize π (ie. $\epsilon - \mathtt{greedy}, \beta$).
2: Initialize $V(s)$ arbitrarily, $e(s) = 0$ for all (s).
3: Return a using $\pi(s)$.
4: Take action a, observe reward r, and next state, s'.
5: $\delta \leftarrow r + \gamma V(s') - V(s)$.
6: $e(s) \leftarrow e(s) + 1$.
7: **for** For all s: **do**
8: $\quad V(s) \leftarrow V(s) + \alpha\delta e(s)$
9: $\quad e(s) \leftarrow \gamma\lambda e(s)$
10: **end for**
11: $s \leftarrow s'$
12: Increment t. If $t \leqslant T$, go to line 3.

TD(λ) then makes use of a learning rate parameter to incrementally learn from the one-step return standard to TD methods, while DQ-C recursively updates either a sample or exponentially weighted average. TD(λ) requires three parameters: a discount factor, γ, a learning rate, α, and a parameter to control the length of the backup, λ.

DQ-C requires only the use of a discount factor, γ, in the sample average case, and adds a learning rate or base weight, α, that serves a similar purpose as λ in TD(λ) in the exponentially weighted case. Several potential strategies are yet to be explored to eliminate the need for α in the exponential version of DQ-C. The consideration of time in the application of weights in the exponentially weighted version of the algorithm also serves to distinguish DQ-C from other approaches based on TD(λ), such as Q(λ) and SARSA(λ). The next section examines the convergence properties of DQ-C.

Analysis of DQ-C

In this section we will analyze the convergence properties of DQ-C in a stationary episodic task environment.

Definition 1. *Let* $\{a_n\}$ *be a sequence of real numbers. We say that* $\{a_n\}$ *is Cauchy convergent provided for every* $\epsilon > 0$*, there is an N(*ϵ*) such that,* $n > N(\epsilon)$ *and* $m > N(\epsilon)$ *implies* $|a_n - a_m| < \epsilon$.

Lemma 1. *Given an episodic task environment, a stochastic* π*, and an infinite number of trials,* $t \to \infty$*, the number of visits,* n*, to each* (s, a) *will go to* ∞ *as will the number of* k *point rewards.*

Assume DQ-C is paired with a stochastic π. This guarantees that as $t \to \infty$, $n \to \infty$ for each (s, a) and since $R(s)$ assigns point rewards based on visits to s we can see that $k \to \infty$ as well. We also know by that $\gamma^{t_i - t_j}$ forms a geometric series since $t_i > t_j \forall i \in k$ for each j, and that as $k \to \infty$ the geometric series $\gamma^{t_i - t_j}$ converges to 0.

Theorem 1. *The estimate of the utility of each* j^{th} *visit to a* (s, a)*,* $U_j(k)$*, converges as* $t \to \infty$.

Assuming constant non-negative rewards, we see that $|U_j(i-1) - U_j(i)| > |U_j(i) - U_j(i+1)| \forall i \in k$ and that as $k \to \infty$, $|U_j(k) - U_j(k+1)| \to 0$, since with fixed t_j, $t_{k+1} - t_j \to \infty$ and $\gamma^\infty \to 0$, and we can see that there exists a $N(\epsilon)$ such that $|U_j(i+n) - U_j(i+m)| < \epsilon, \forall n, m > N(\epsilon)$.

Theorem 2. *The estimate of the sample average of each* (s, a)*,* $Q(n, k)$*, converges to the true mean value of a visit to* (s, a)*,* μ*, as t*$\to \infty$.

$Q(n, k)$ can be viewed as the sample mean of a convergent series of partial sums by examining the discussion of $U_j(k)$ and we can leverage the convergent properties of $U_j(k)$ to show that the right hand side of the one-step update of $Q(n, k)$ goes to zero as $n, k \to \infty$, since $\lim_{n,k\to\infty} \gamma^{t_{k+1} - t_j} = 0$, and since $\lim_{n\to\infty} \frac{n}{n+1} = 1$ we see that $\lim_{n,k\to\infty} \frac{n}{n+1} Q(n, k) = Q(n, k)$.

Theorem 3. *The estimate of the exponentially weighted average of each* (s, a)*,* $Q(n, k)$*, converges to the recency weighted mean of a visit to* (s, a)*,* $\mu(n, k, t)$*, as* $t, n, k \to \infty$.

The exponentially weighted average provides an estimate of the current value of a (s, a) by ensuring that as the $t \to \infty$ that with fixed t_j, $(t - t_j) \to \infty$ and that the weight, $\alpha^{t-t_j} \to 0$, for $0 < \alpha < 1$. We can see that since the weights are normalized that as initial weights tend toward 0, greater emphasis is placed on new observations and that for $|Q(n, k, t) - Q(n, k, t + 1)| < \epsilon$ as $t \to \infty$ for fixed n, k and that $\alpha^{t-t_n} < \alpha^{t-t_{n+1}} \forall j \in n$. We see that as the interval $t - t_n \to \infty$ that the estimate converges to the $n + 1^{th}$ observation.

$$\begin{aligned} \lim_{t,n,k\to\infty} Q(n, k, t) &= \frac{\alpha^t S(n, k) + \alpha^{t-t_{n+1}} u_{n+1}(k)}{\alpha^t C(n) + \alpha^{t-t_{n+1}}} \\ &= \frac{\alpha^{t-t_{n+1}} u_{n+1}(k)}{\alpha^{t-t_{n+1}}} = u_{n+1}(k) \end{aligned} \tag{87}$$

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX E: PTSP MAPS

Map sets for the Physical Traveling Salesman Problem.

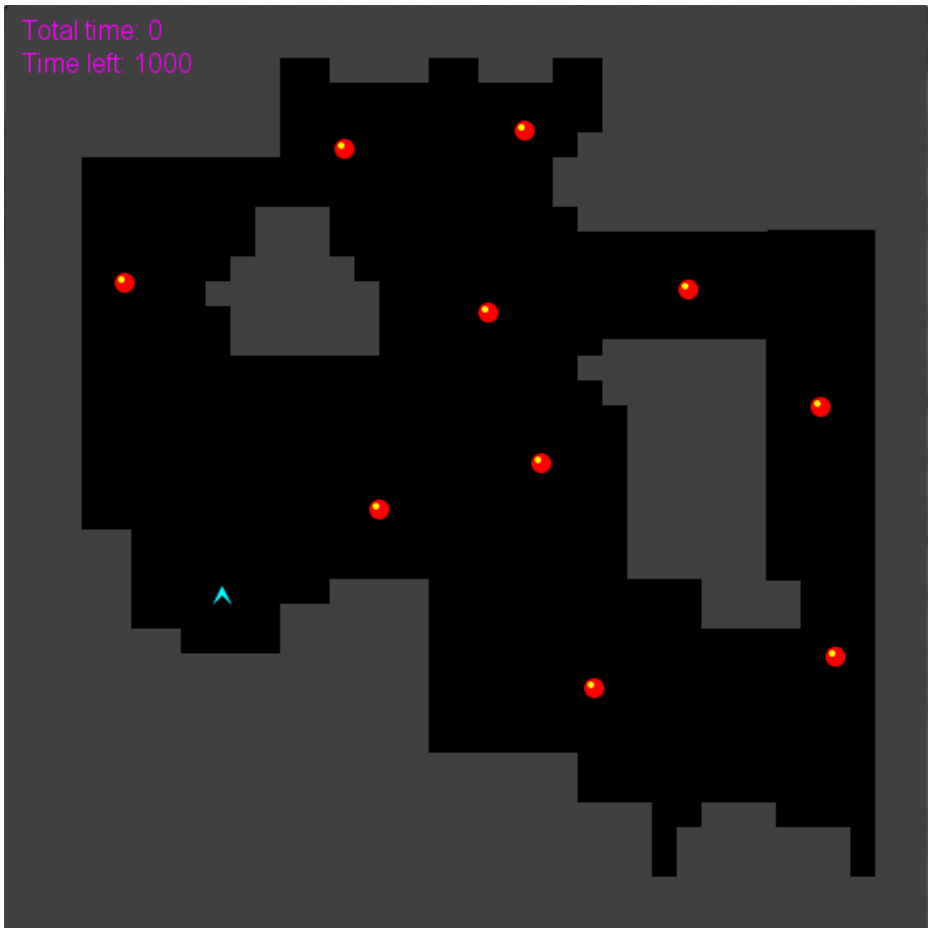


Figure 45: Map 2 for physical traveling salesman problem.

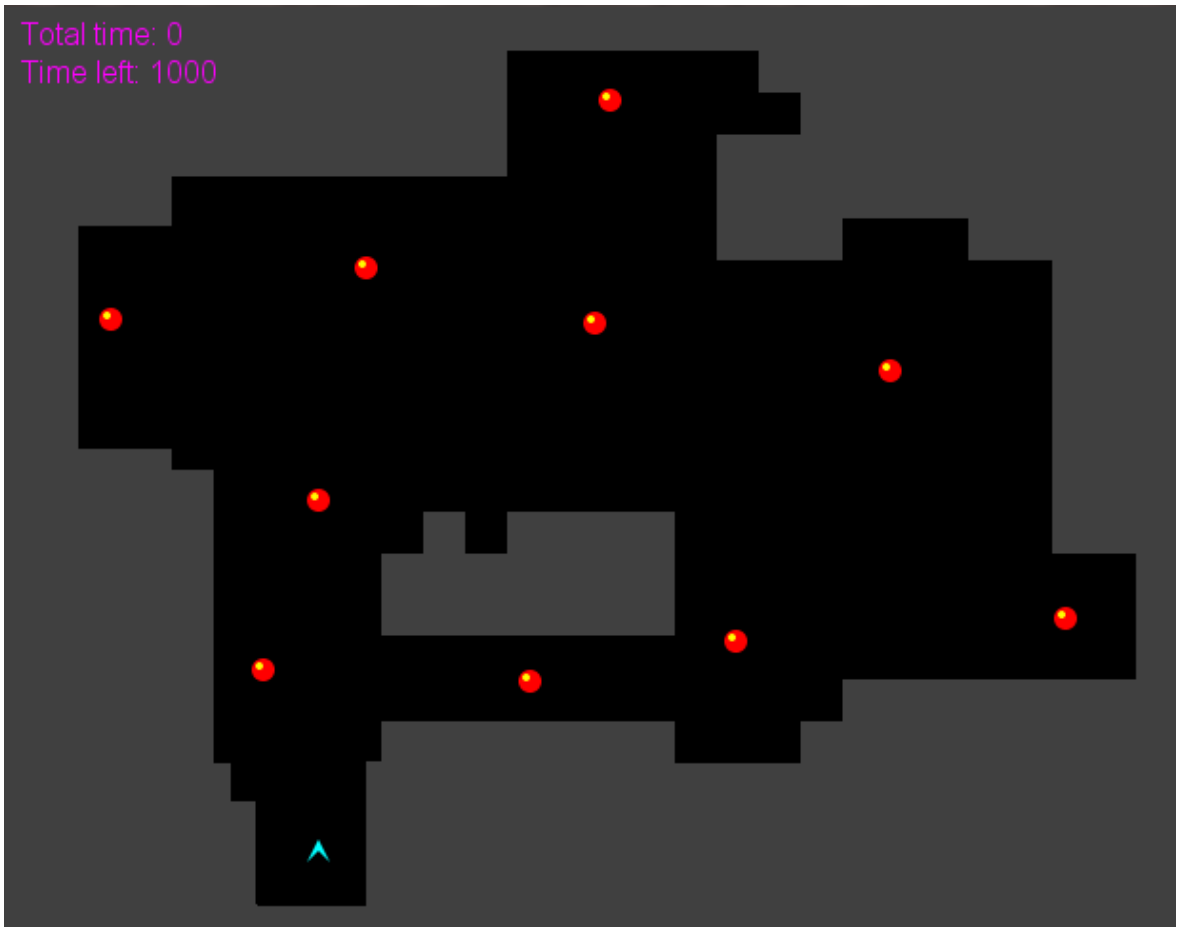


Figure 46: Map 3 for physical traveling salesman problem.

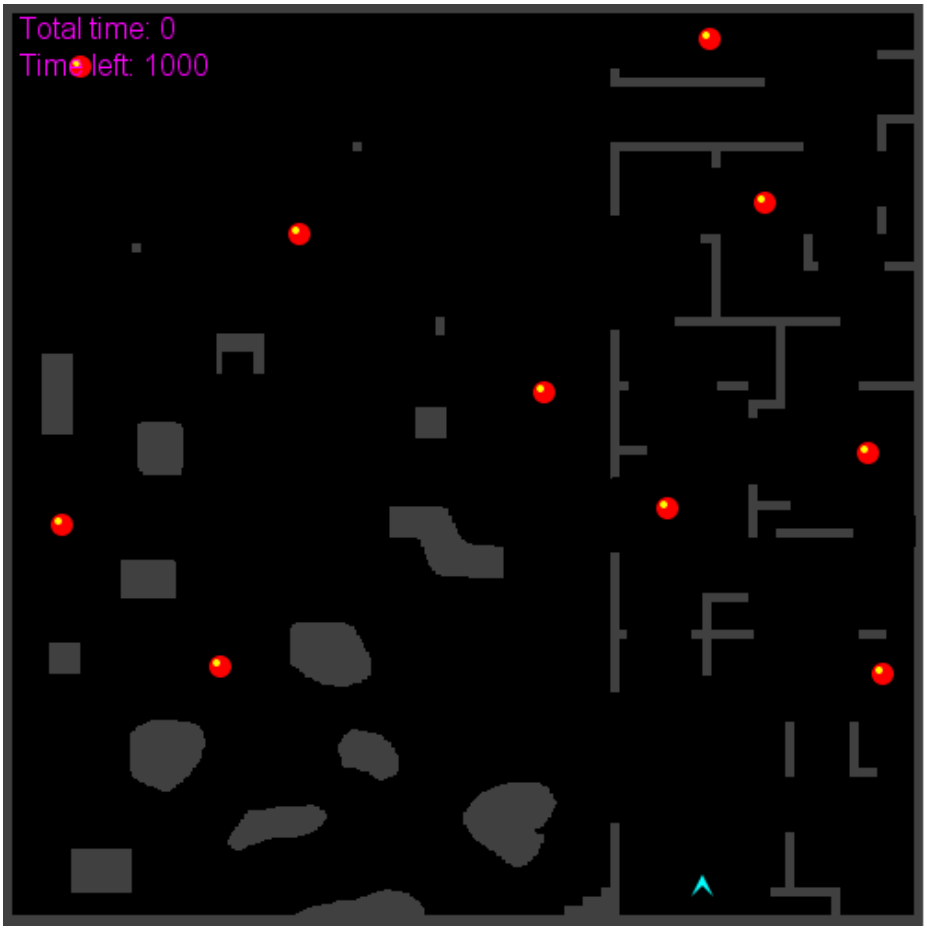


Figure 47: Map 4 for physical traveling salesman problem.

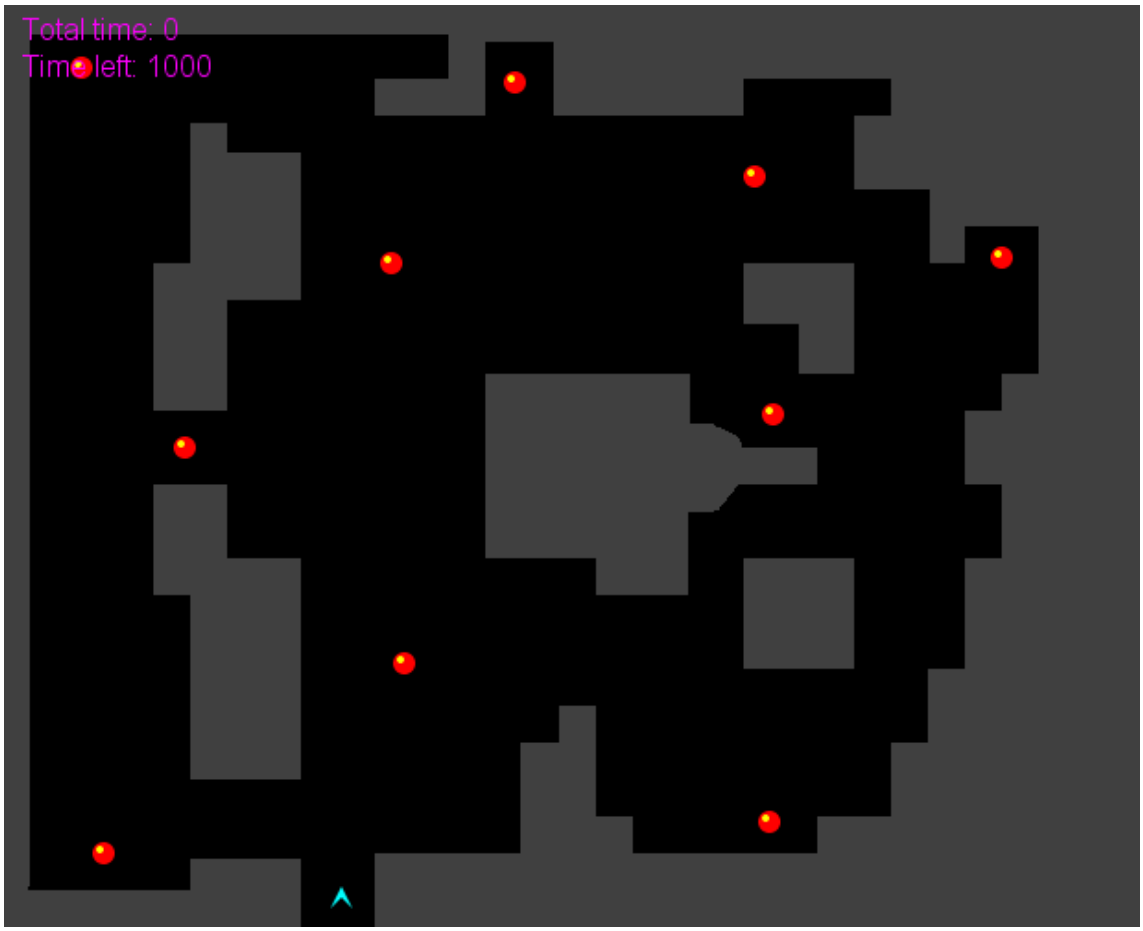


Figure 48: Map 5 for physical traveling salesman problem.

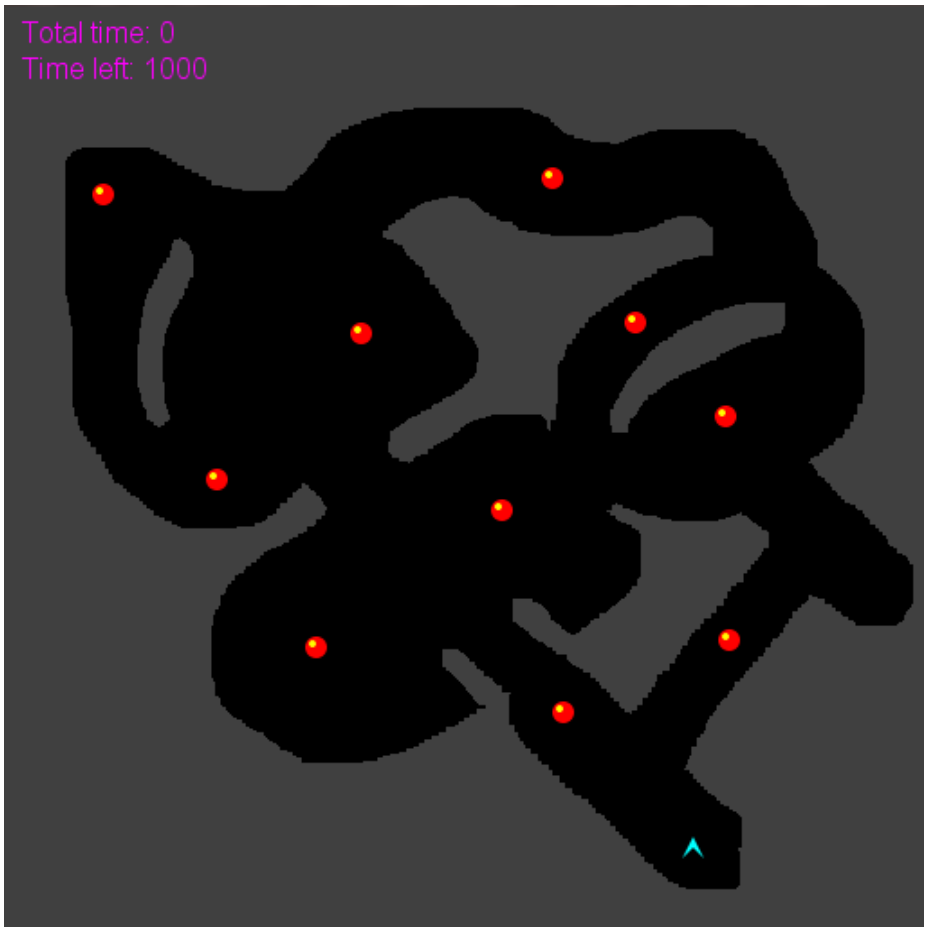


Figure 49: Map 6 for physical traveling salesman problem.

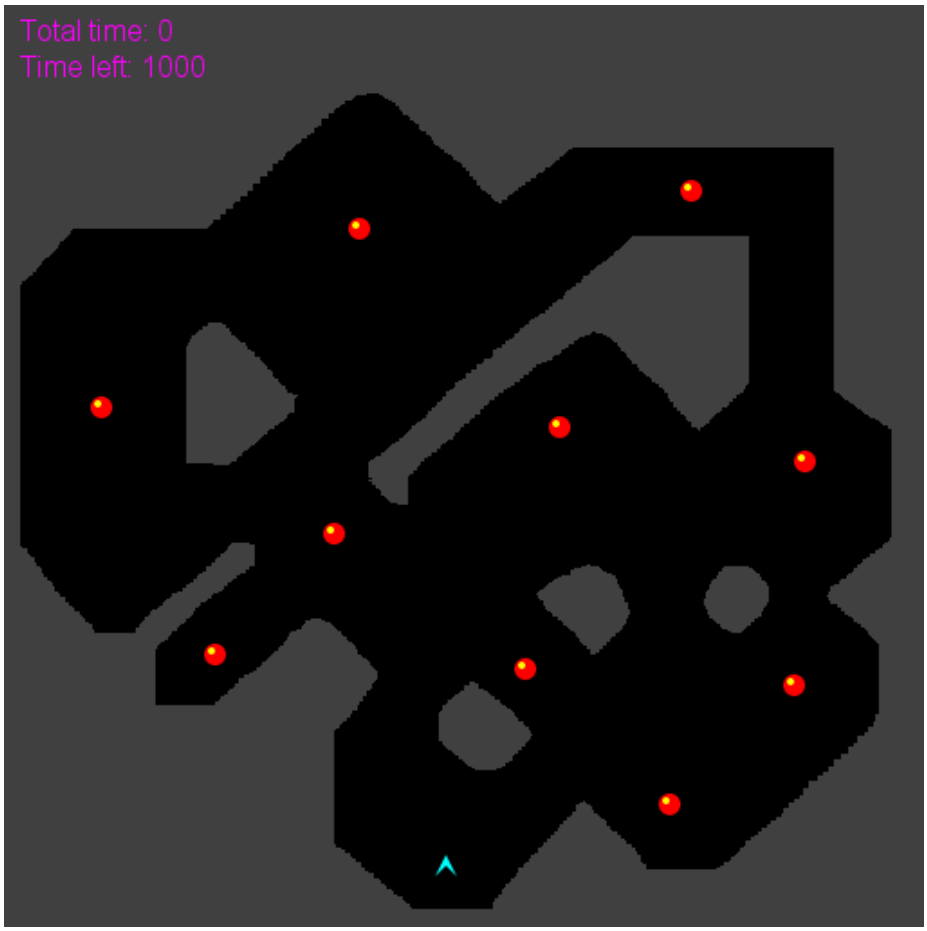


Figure 50: Map 7 for physical traveling salesman problem.

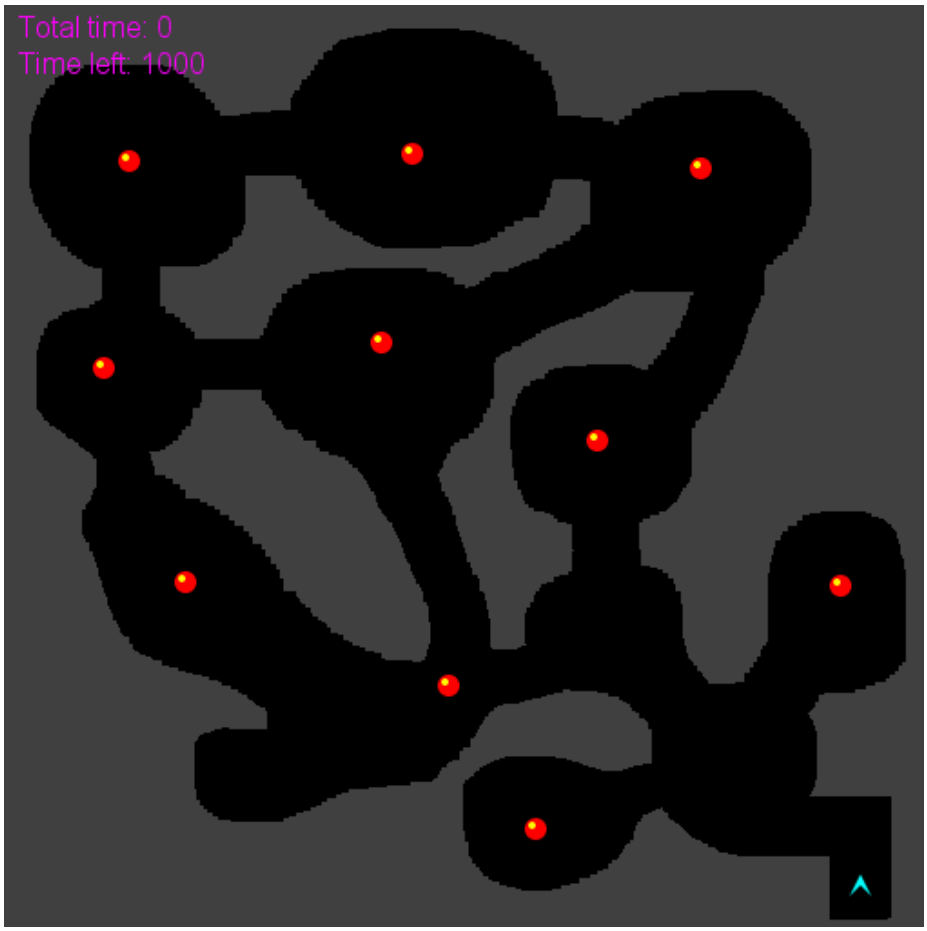


Figure 51: Map 8 for physical traveling salesman problem.

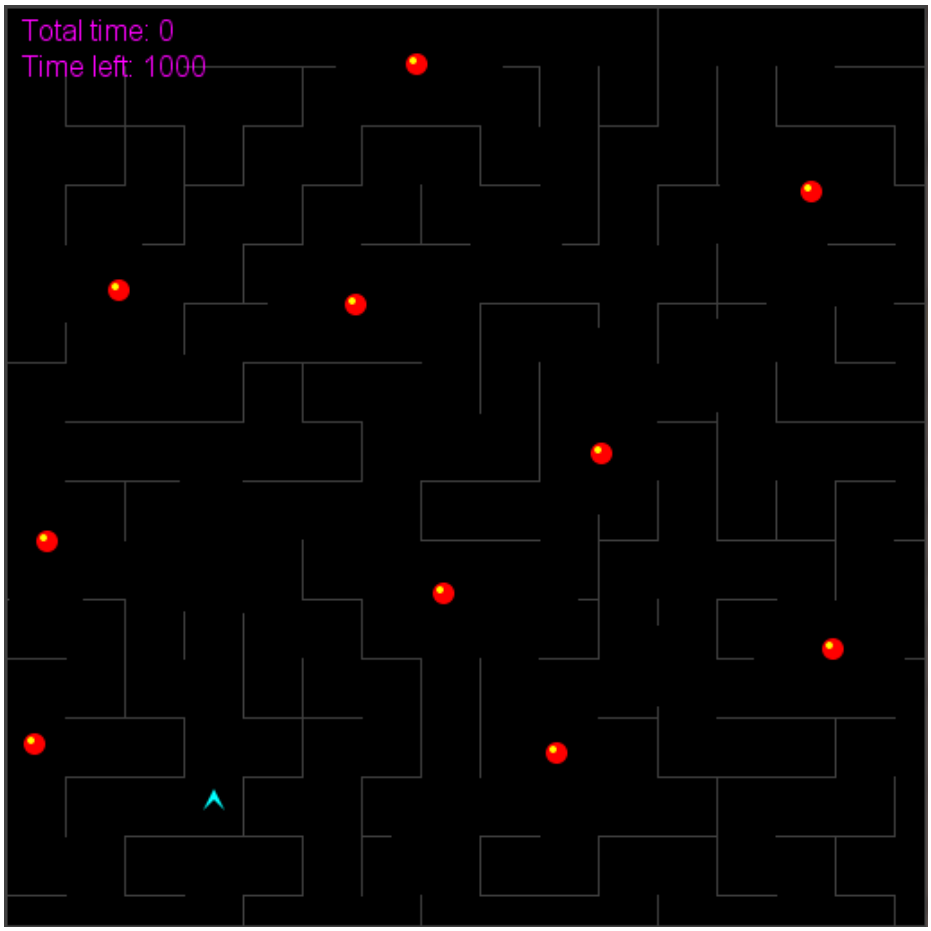


Figure 52: Map 9 for physical traveling salesman problem.

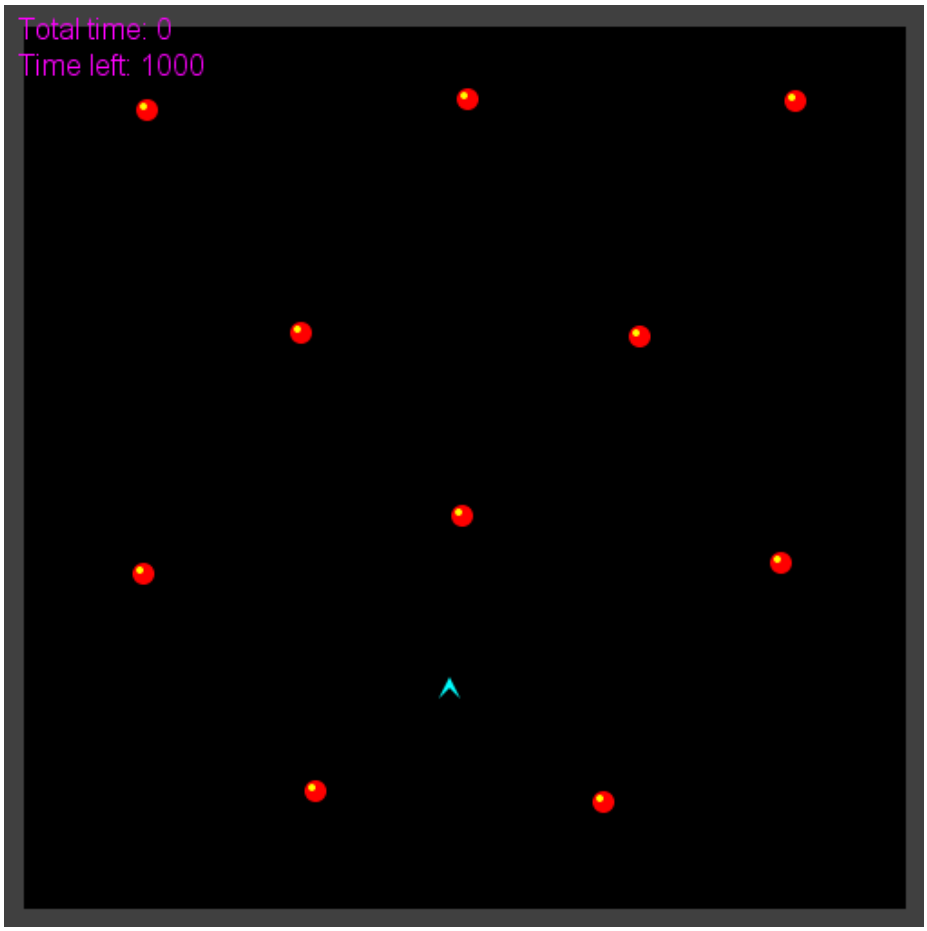


Figure 53: Map 10 for physical traveling salesman problem.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX F: FULL ASC-U FORMULATION

ASC-U formulates this problem as a finite state and finite time horizon dynamic program with the following indices (Ahner et al., 2006),

- t, current time, $t = 0, 1, ...T$, where T maximum simulation time,
- i, i^{th} UAV, $i \in UAV$, where UAV is the set of UAV,
- j, j^{th} GCS, $j \in GCS$, where GCS is the set of GCS,
- k, k^{th} LRS, $k \in LRS$, where LRS is the set of LRS,
- l, l^{th} Mission, $l \in Msn$, where Msn is the set of Missions,

and the following additional terms,

- τ, the time horizon over which the system is optimized,
- $\langle x, y \rangle_t$, the cartesian coordinate of an object at t,
- $t_{flight_t}^{UAV_i}$, time of flight of i^{th} UAV at t
- $\overrightarrow{u_t}$, a decision vector that acts on the system selected from the finite set U at each t,
- $\overrightarrow{s_t}$, the vector describing the state of the system at t, where $\overrightarrow{s_t} \in S$,
- $\overrightarrow{s_{t+\tau}}$, the state of the system at the end of the next time horizon,

used in the dynamic programming formulation where the update to $\overrightarrow{s_{t+\tau}}$ is further defined,

$$\overrightarrow{s_{t+\tau}} = f_1\left(\overrightarrow{s_t}, \overrightarrow{u_t}, \tau\right) \tag{88}$$

and $\overrightarrow{s_t}$ is further defined,

$$
\begin{aligned}
\overrightarrow{s_t} &\equiv \left(s_t^{UAV_i}, s_t^{GCS_j}, s_t^{LRS_k}, s_t^{Msn_l} \right), \\
\overrightarrow{s_t^{UAV_i}} &\equiv \left(\langle x, y \rangle_t^{UAV_i}, t_{flight}^{UAV_i}, t_{recovery}^{UAV_i}, A_t^{UAV_i}, sensor_t^{UAV_i}, type^{UAV_i} \right), \\
\overrightarrow{s_t^{GCS_j}} &\equiv \left(\langle x, y \rangle_t^{GCS_j}, capacity_t^{GCS_j}, type^{GCS_j} \right), \\
\overrightarrow{s_t^{LRS_k}} &\equiv \left(\langle x, y \rangle_t^{LRS_k}, capacity_t^{LRS_k}, type^{LRS_k} \right), \\
\overrightarrow{s_t^{Msn_l}} &\equiv \left(\langle x, y \rangle_t^{Msn_l}, sensor_m \right),
\end{aligned}
\tag{89}
$$

where,

$$
sensor_m \equiv \left(sensorType_m, \langle t_open, t_endOpen \rangle, v_rate \right), \tag{90}
$$

describes the seor requirement of each mission area.

In the dynamic programming formulation the cost-to-go or future value function at time t is used to determine the optimal policy, here mapping $\pi(s) \rightarrow U$,

$$
J_t(s_t) = \max_{\overrightarrow{u_t}} C_t(s_t, \overrightarrow{u_t}) + J_{t+\tau}(s_{t+\tau}), \tag{91}
$$

for $t = 0...T - \tau_f$, where τ_f is the time difference from the end of the time horizon and the last applied control. Since for the deterministic case all mission demands are known in advance the optimal control policy could be obtained by,

$$
u_t = \max_{\overrightarrow{u_t}} C_t(s_t, \overrightarrow{u_t}) + J_{t+\tau}(s_{t+\tau}), \tag{92}
$$

The problem as addressed in ASC-U is formulated using the following indices (Ahner et al., 2006),

- t, current time, $t = 0, 1, ...T$, where T maximum simulation time,
- i, i^{th} UAV, $i \in UAV$, where UAV is the set of UAV,
- j, j^{th} GCS, $j \in GCS$, where GCS is the set of GCS,
- k, k^{th} LRS, $k \in LRS$, where LRS is the set of LRS,
- l, l^{th} Mission, $l \in Msn$, where Msn is the set of Missions,

and the following sets,

- A, set of all mission areas with active missions during the optimization time horizon,
- L, set of all active LRS,
- G, set of all GCS,
- G_L, set of all GCS assigned to LRS L,
- C_g, number of UAV GCS g is capable of controlling,
- I_t, sub-set of UAV at LRS $l \in L$, defined by min $\Big($ready UAV, launch limit, airborne UAV,$\sum_{G_L} C_g$m assigned UAV to GCS$\Big)$
- J_l, subset of all sensor packages located at LRS, $l \in L$,
- $Y_{ga} = 1$, if mission area a assigned to GCS g, 0 otherwise,
- c_{ja}, reward for a UAV with sensor package j being assigned to mission area a from the soonest possible arrival time of the UAV at a to the end of the time horizon, $t + \delta t^i$, for UAV_i,
- $X_{ja} = 1$, if a UAV with sensor package j is assigned to a and 0 otherwise.

The formulation for this revised problem seeks to maximize the value of the mission areas covered subject to four constraints,

$$\max \sum_{ja} c_{ja} X_{ja} \tag{93}$$

subject to,

$$\sum_{j} X_{ja} \leqslant 1 \forall a \in A \tag{94}$$

, ensures only a single UAV to a mission area,

$$\sum_{a} X_{ja} \leqslant 1 \forall i \in I \tag{95}$$

, ensures only a single mission area is assigned to each UAV,

$$\sum_{ja} Y_{ga} X_{ja} + g \forall g \in G \tag{96}$$

, constrains the number of UAV's a single GCS can control and,

$$\sum_{j \in J_{la}} X_{ja} \leqslant |I_l| \forall l \in L \tag{97}$$

, ensures that the sensors assigned to an area does not exceed the capacity of the assigned UAVs.

The assignment of mission areas to GCS is determined by a heuristic, see Algorithm 15, as is the determination of the reward, see Algorithms 16. This reformulation proved successful in application and was successfully applied to several real-world studies by both TRAC and MCCDC-OAD. We discuss the application of RL to this problem in the following section.

Algorithm 15 Assignment of Mission Areas to GCS.

for each LRS and mission area a **do**
  $N_a \equiv$ the number of GCS_g that are in range of UAV assigned to a.
end for
for each mission area a **do**
  sort by N_a.
  for each GCS_g **do**
    sort by $\sum_a Y_{ga}$
    If a is in range of g, let $Y_{ga} = 1$
end for

Algorithm 16 Assignment of value for completed missions.

for each UAV i **do**
  for each mission area a **do**
    $t_0 =$ first time after the earliest arrival time that UAV i can gain value by being assigned to mission area a
    $t_1 = \min\left(\text{the latest time UAV } i \text{ can remain at mission area } a,\ t + \delta t^i, \text{ for UAV } i\right)$
    $K_a =$ the set of all missions located at mission area a
    $V_{i,k,t_0,t_1} =$ the value that UAV i gains from mission k by being at mission area a
    Calculate, $c_i a = \sum_{k_a} V_{i,k,t_o,t_1}$
  end for
end for

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX G: LEARNED STATE-ACTION VALUES FOR URBANSIM.

Table 20: Learned policy for Battalion Commander by turn for 15 turn UrbanSim game using DQ-C.

STRATEGY	1	2	3	4	5	6	7	8
SKC	6.6E-316	6.6E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
SKH	2.3E+02	2.4E+02	2.4E+02	2.4E+02	2.4E+02	2.4E+02	2.4E+02	2.5E+02
SKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
SNC	2.4E+02	2.4E+02	2.3E+02	2.3E+02	2.4E+02	2.4E+02	2.4E+02	2.4E+02
SNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
SNB	2.3E+02	2.6E+02	2.3E+02	2.4E+02	2.3E+02	2.4E+02	2.4E+02	2.5E+02
MKC	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
MKH	2.6E+02	2.3E+02	2.6E+02	2.4E+02	2.4E+02	2.4E+02	2.4E+02	2.3E+02
MKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
MNC	2.4E+02	2.4E+02	2.4E+02	2.6E+02	2.3E+02	2.4E+02	2.4E+02	2.4E+02
MNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
MNB	0.0E+00 9	2.4E+02 10	2.3E+02 11	2.3E+02 12	2.4E+02 13	2.4E+02 14	2.4E+02 15	2.4E+02
SKC	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
SKH	2.4E+02	2.5E+02	2.5E+02	2.6E+02	2.5E+02	2.6E+02	2.6E+02	
SKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
SNC	2.5E+02	2.5E+02	2.6E+02	2.6E+02	2.6E+02	2.6E+02	3.0E+02	
SNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
SNB	2.4E+02	2.5E+02	2.6E+02	2.9E+02	2.5E+02	2.6E+02	2.6E+02	
MKC	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
MKH	2.5E+02	2.4E+02	2.9E+02	2.6E+02	2.6E+02	2.6E+02	2.6E+02	
MKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
MNC	2.5E+02	2.8E+02	2.5E+02	2.6E+02	2.6E+02	2.6E+02	2.6E+02	
MNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
MNB	2.5E+02	2.5E+02	2.5E+02	2.5E+02	2.6E+02	0.0E+00	2.6E+02	

Table 21: Learned policy for CA unit by turn for 15 turn UrbanSim game using DQ-C.

STRATEGY	1	2	3	4	5	6	7	8
SKC	6.6E-316	6.6E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
SKH	2.4E+02	2.3E+02	2.4E+02	2.3E+02	2.3E+02	2.4E+02	2.4E+02	2.5E+02
SKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
SNC	2.6E+02	2.4E+02	2.3E+02	2.4E+02	2.3E+02	2.4E+02	2.7E+02	2.5E+02
SNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
SNB	2.4E+02	2.4E+02	2.3E+02	2.3E+02	2.4E+02	2.4E+02	2.4E+02	2.4E+02
MKC	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
MKH	2.3E+02	2.6E+02	2.6E+02	2.4E+02	2.3E+02	2.4E+02	2.3E+02	2.4E+02
MKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
MNC	2.3E+02	2.4E+02	0.0E+00	2.4E+02	2.4E+02	2.3E+02	2.4E+02	2.4E+02
MNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
MNB	2.4E+02	2.4E+02	2.4E+02	2.3E+02	2.4E+02	2.4E+02	0.0E+00	2.4E+02
	9	10	11	12	13	14	15	
SKC	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
SKH	2.5E+02	2.8E+02	2.6E+02	2.6E+02	0.0E+00	2.7E+02	2.6E+02	
SKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
SNC	2.4E+02	2.5E+02	2.5E+02	2.6E+02	2.6E+02	2.6E+02	2.7E+02	
SNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
SNB	2.4E+02	2.5E+02	2.6E+02	2.6E+02	2.6E+02	2.6E+02	2.6E+02	
MKC	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
MKH	2.4E+02	2.5E+02	2.6E+02	2.6E+02	2.5E+02	3.0E+02	2.6E+02	
MKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
MNC	2.8E+02	2.5E+02	2.5E+02	2.6E+02	2.6E+02	2.6E+02	2.6E+02	
MNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
MNB	2.5E+02	2.5E+02	2.5E+02	2.5E+02	0.0E+00	2.6E+02	2.6E+02	

Table 22: Learned policy for E CO a by turn for 15 turn UrbanSim game using DQ-C.

STRATEGY	1	2	3	4	5	6	7	8
SKC	6.3E-316	4.8E-316	1.0E-315	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316
SKH	2.6E+02	2.4E+02	2.6E+02	2.3E+02	2.4E+02	2.4E+02	2.3E+02	2.5E+02
SKB	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316
SNC	2.4E+02	2.4E+02	2.3E+02	2.3E+02	2.3E+02	2.4E+02	2.4E+02	2.4E+02
SNH	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316
SNB	2.4E+02	2.3E+02	2.4E+02	2.4E+02	0.0E+00	2.4E+02	2.4E+02	2.4E+02
MKC	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316
MKH	2.3E+02	2.4E+02	2.3E+02	2.4E+02	2.3E+02	2.4E+02	2.4E+02	2.4E+02
MKB	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316
MNC	2.4E+02	2.3E+02	2.3E+02	2.4E+02	2.3E+02	2.3E+02	2.7E+02	2.3E+02
MNH	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316
MNB	2.4E+02	2.4E+02	2.4E+02	2.3E+02	2.6E+02	2.4E+02	2.4E+02	2.4E+02
	9	10	11	12	13	14	15	
SKC	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	
SKH	2.5E+02	0.0E+00	2.5E+02	2.6E+02	2.5E+02	2.7E+02	3.0E+02	
SKB	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	
SNC	2.4E+02	2.5E+02	2.5E+02	2.5E+02	2.6E+02	2.6E+02	0.0E+00	
SNH	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	
SNB	2.4E+02	2.5E+02	2.6E+02	2.6E+02	2.5E+02	2.6E+02	2.6E+02	
MKC	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	
MKH	2.5E+02	2.5E+02	2.6E+02	2.9E+02	2.6E+02	2.6E+02	2.5E+02	
MKB	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	
MNC	2.5E+02	2.5E+02	2.9E+02	2.6E+02	2.6E+02	2.6E+02	2.6E+02	
MNH	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	
MNB	2.8E+02	2.5E+02	2.6E+02	2.5E+02	2.6E+02	2.6E+02	2.7E+02	

Table 23: Learned policy for F CO a by turn for 15 turn UrbanSim game using DQ-C.

STRATEGY	1	2	3	4	5	6	7	8
SKC	6.3E-316	4.8E-316	1.0E-315	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316
SKH	0.0E+00	2.3E+02	2.3E+02	2.4E+02	2.4E+02	2.4E+02	2.4E+02	2.4E+02
SKB	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316
SNC	2.4E+02	0.0E+00	2.4E+02	2.4E+02	2.4E+02	2.4E+02	2.7E+02	2.4E+02
SNH	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316
SNB	2.6E+02	2.4E+02	2.4E+02	2.3E+02	2.3E+02	2.4E+02	2.4E+02	2.7E+02
MKC	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316
MKH	2.4E+02	2.4E+02	2.3E+02	2.4E+02	2.3E+02	2.4E+02	2.4E+02	2.4E+02
MKB	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316
MNC	2.3E+02	2.3E+02	2.4E+02	2.4E+02	2.4E+02	2.4E+02	2.4E+02	2.5E+02
MNH	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316
MNB	2.4E+02	2.4E+02	2.3E+02	2.3E+02	2.3E+02	2.4E+02	2.4E+02	2.4E+02
	9	10	11	12	13	14	15	
SKC	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	
SKH	2.4E+02	2.5E+02	2.6E+02	2.6E+02	2.6E+02	3.0E+02	2.6E+02	
SKB	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	
SNC	2.4E+02	2.5E+02	2.5E+02	2.9E+02	2.6E+02	2.6E+02	2.7E+02	
SNH	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	
SNB	2.5E+02	2.5E+02	2.5E+02	2.5E+02	2.6E+02	2.7E+02	2.6E+02	
MKC	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	
MKH	2.5E+02	2.5E+02	2.6E+02	2.6E+02	2.6E+02	2.6E+02	2.6E+02	
MKB	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	
MNC	2.5E+02	2.5E+02	2.9E+02	2.6E+02	2.9E+02	0.0E+00	3.0E+02	
MNH	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	
MNB	2.4E+02	2.5E+02	2.5E+02	0.0E+00	2.5E+02	2.6E+02	2.6E+02	

Table 24: Learned policy for E CO b by turn for 15 turn UrbanSim game using DQ-C.

STRATEGY	1	2	3	4	5	6	7	8
SKC	6.3E-316	4.8E-316	1.0E-315	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316
SKH	2.4E+02	2.4E+02	2.4E+02	2.4E+02	2.3E+02	2.4E+02	2.4E+02	2.4E+02
SKB	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316
SNC	2.3E+02	2.3E+02	2.6E+02	2.3E+02	2.3E+02	2.4E+02	2.4E+02	2.5E+02
SNH	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316
SNB	2.6E+02	2.3E+02	2.3E+02	2.4E+02	2.4E+02	2.6E+02	2.4E+02	2.5E+02
MKC	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316
MKH	2.3E+02	2.4E+02	2.3E+02	2.3E+02	2.4E+02	2.4E+02	2.4E+02	2.3E+02
MKB	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316
MNC	2.4E+02	2.4E+02	2.3E+02	2.4E+02	2.4E+02	2.4E+02	2.4E+02	2.5E+02
MNH	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316
MNB	2.4E+02 9	2.4E+02 10	2.3E+02 11	2.4E+02 12	2.4E+02 13	2.4E+02 14	2.4E+02 15	2.4E+02
SKC	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	
SKH	2.5E+02	2.5E+02	2.5E+02	2.9E+02	2.6E+02	0.0E+00	2.6E+02	
SKB	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	
SNC	2.8E+02	2.8E+02	2.5E+02	2.6E+02	2.6E+02	2.7E+02	2.6E+02	
SNH	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	
SNB	2.5E+02	2.5E+02	2.9E+02	2.6E+02	2.6E+02	2.6E+02	2.6E+02	
MKC	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	
MKH	2.4E+02	2.5E+02	2.6E+02	2.6E+02	0.0E+00	2.6E+02	0.0E+00	
MKB	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	
MNC	2.4E+02	2.5E+02	2.6E+02	2.5E+02	2.6E+02	2.6E+02	2.6E+02	
MNH	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	
MNB	2.4E+02	2.5E+02	2.6E+02	2.6E+02	2.6E+02	2.6E+02	3.0E+02	

Table 25: Learned policy for F CO b by turn for 15 turn UrbanSim game using DQ-C.

STRATEGY	1	2	3	4	5	6	7	8
SKC	6.3E-316	4.8E-316	1.0E-315	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316
SKH	2.4E+02	2.6E+02	2.3E+02	2.4E+02	2.4E+02	2.4E+02	2.4E+02	2.4E+02
SKB	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316
SNC	2.3E+02	2.4E+02	0.0E+00	0.0E+00	2.6E+02	2.4E+02	2.4E+02	2.7E+02
SNH	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316
SNB	2.4E+02	2.3E+02	2.3E+02	2.3E+02	2.3E+02	2.3E+02	2.4E+02	2.4E+02
MKC	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316
MKH	2.4E+02	2.4E+02	2.4E+02	2.3E+02	2.3E+02	2.4E+02	2.4E+02	2.3E+02
MKB	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316
MNC	2.4E+02	2.4E+02	2.3E+02	2.4E+02	2.4E+02	2.4E+02	2.4E+02	0.0E+00
MNH	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316
MNB	2.3E+02	2.4E+02	2.3E+02	2.3E+02	2.4E+02	2.4E+02	2.4E+02	2.5E+02
	9	10	11	12	13	14	15	
SKC	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	
SKH	2.4E+02	2.5E+02	2.5E+02	2.6E+02	2.6E+02	2.6E+02	2.6E+02	
SKB	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	
SNC	2.8E+02	2.5E+02	2.6E+02	2.9E+02	2.6E+02	0.0E+00	2.5E+02	
SNH	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	
SNB	2.5E+02	2.5E+02	2.6E+02	2.6E+02	2.6E+02	2.6E+02	2.6E+02	
MKC	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	
MKH	2.5E+02	2.5E+02	2.6E+02	2.6E+02	2.6E+02	2.6E+02	2.6E+02	
MKB	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	
MNC	2.5E+02	2.8E+02	2.5E+02	2.6E+02	2.6E+02	2.7E+02	2.6E+02	
MNH	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	
MNB	0.0E+00	2.5E+02	2.5E+02	2.5E+02	2.6E+02	2.7E+02	2.6E+02	

Table 26: Learned policy for G CO a by turn for 15 turn UrbanSim game using DQ-C.

STRATEGY	1	2	3	4	5	6	7	8
SKC	6.5E-316	4.8E-316	1.0E-315	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316
SKH	2.4E+02	2.4E+02	2.3E+02	2.4E+02	2.4E+02	2.4E+02	2.3E+02	2.4E+02
SKB	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316
SNC	2.3E+02	0.0E+00	2.4E+02	2.3E+02	2.3E+02	2.4E+02	2.4E+02	2.3E+02
SNH	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316
SNB	2.6E+02	0.0E+00	2.3E+02	2.3E+02	2.3E+02	2.4E+02	2.4E+02	0.0E+00
MKC	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316
MKH	2.4E+02	2.3E+02	2.4E+02	2.4E+02	2.4E+02	2.4E+02	2.4E+02	2.4E+02
MKB	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316
MNC	2.4E+02	2.4E+02	2.6E+02	2.3E+02	2.4E+02	2.4E+02	2.4E+02	2.5E+02
MNH	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316
MNB	2.3E+02	2.4E+02	0.0E+00	2.4E+02	2.3E+02	2.4E+02	2.4E+02	2.4E+02
	9	10	11	12	13	14	15	
SKC	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	
SKH	2.5E+02	2.5E+02	2.5E+02	2.6E+02	2.5E+02	2.6E+02	2.6E+02	
SKB	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	
SNC	2.4E+02	2.5E+02	0.0E+00	2.6E+02	2.6E+02	2.6E+02	2.6E+02	
SNH	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	
SNB	2.5E+02	2.5E+02	2.5E+02	2.6E+02	2.6E+02	2.7E+02	3.0E+02	
MKC	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	
MKH	2.5E+02	0.0E+00	2.6E+02	2.6E+02	2.6E+02	2.6E+02	2.5E+02	
MKB	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	
MNC	2.5E+02	2.5E+02	2.5E+02	2.5E+02	2.6E+02	3.0E+02	0.0E+00	
MNH	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	
MNB	2.4E+02	2.5E+02	2.5E+02	2.6E+02	2.6E+02	2.7E+02	2.6E+02	

Table 27: Learned policy for H CO a by turn for 15 turn UrbanSim game using DQ-C.

STRATEGY	1	2	3	4	5	6	7	8
SKC	6.3E-316	4.8E-316	2.3E+02	2.3E+02	2.4E+02	2.4E+02	2.7E+02	2.4E+02
SKH	2.4E+02	2.4E+02	0.0E+00	2.4E+02	2.6E+02	2.4E+02	2.7E+02	2.3E+02
SKB	2.4E+02	2.4E+02	0.0E+00	2.4E+02	2.3E+02	2.4E+02	2.4E+02	2.5E+02
SNC	2.4E+02	2.4E+02	2.3E+02	2.3E+02	2.4E+02	2.4E+02	2.4E+02	2.4E+02
SNH	2.4E+02	2.4E+02	2.4E+02	2.3E+02	2.6E+02	2.4E+02	2.4E+02	2.5E+02
SNB	2.4E+02	2.3E+02	2.3E+02	2.6E+02	2.4E+02	2.4E+02	0.0E+00	2.7E+02
MKC	2.4E+02	2.3E+02	2.4E+02	2.4E+02	2.3E+02	2.6E+02	2.4E+02	2.4E+02
MKH	2.3E+02	2.4E+02	2.4E+02	2.4E+02	2.4E+02	2.4E+02	2.4E+02	2.4E+02
MKB	2.3E+02	2.4E+02	2.3E+02	2.4E+02	0.0E+00	2.3E+02	2.4E+02	2.4E+02
MNC	2.4E+02	2.4E+02	2.3E+02	0.0E+00	2.3E+02	2.4E+02	2.4E+02	2.4E+02
MNH	2.4E+02	2.4E+02	2.3E+02	2.6E+02	0.0E+00	2.4E+02	2.4E+02	2.7E+02
MNB	2.4E+02	2.3E+02	2.3E+02	0.0E+00	2.4E+02	2.4E+02	2.4E+02	2.4E+02
	9	10	11	12	13	14	15	
SKC	2.5E+02	0.0E+00	2.5E+02	2.9E+02	2.9E+02	2.6E+02	2.7E+02	
SKH	2.5E+02	2.5E+02	2.6E+02	2.6E+02	2.6E+02	3.0E+02	2.6E+02	
SKB	2.5E+02	2.5E+02	2.6E+02	2.5E+02	2.6E+02	2.6E+02	2.6E+02	
SNC	2.4E+02	2.5E+02	2.5E+02	2.5E+02	0.0E+00	2.7E+02	3.0E+02	
SNH	2.5E+02	2.5E+02	2.5E+02	2.6E+02	2.6E+02	3.0E+02	3.0E+02	
SNB	2.4E+02	2.5E+02	2.6E+02	2.9E+02	2.6E+02	2.6E+02	2.6E+02	
MKC	2.4E+02	2.8E+02	2.6E+02	2.5E+02	2.6E+02	2.6E+02	2.6E+02	
MKH	2.5E+02	2.8E+02	2.5E+02	2.6E+02	2.6E+02	2.6E+02	2.6E+02	
MKB	2.5E+02	2.5E+02	2.5E+02	2.6E+02	2.6E+02	2.7E+02	2.5E+02	
MNC	2.5E+02	2.5E+02	2.6E+02	2.6E+02	2.5E+02	2.7E+02	2.6E+02	
MNH	2.4E+02	2.5E+02	2.5E+02	2.6E+02	2.6E+02	2.6E+02	2.6E+02	
MNB	2.5E+02	2.5E+02	2.5E+02	2.5E+02	2.5E+02	2.6E+02	2.5E+02	

Table 28: Learned policy for G CO b by turn for 15 turn UrbanSim game using DQ-C.

STRATEGY	1	2	3	4	5	6	7	8
SKC	6.3E-316	4.8E-316	1.0E-315	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316
SKH	2.4E+02	2.4E+02	2.4E+02	2.4E+02	2.3E+02	2.4E+02	2.4E+02	2.4E+02
SKB	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316
SNC	2.3E+02	2.4E+02	2.3E+02	2.4E+02	2.3E+02	2.4E+02	2.4E+02	2.4E+02
SNH	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316
SNB	2.6E+02	2.4E+02	2.6E+02	2.4E+02	2.4E+02	2.4E+02	2.4E+02	2.4E+02
MKC	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316
MKH	2.4E+02	2.4E+02	2.4E+02	2.4E+02	2.4E+02	0.0E+00	2.4E+02	2.5E+02
MKB	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316
MNC	2.4E+02	2.6E+02	2.3E+02	2.4E+02	2.4E+02	2.4E+02	2.4E+02	2.3E+02
MNH	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316
MNB	2.3E+02 9	2.4E+02 10	2.3E+02 11	2.3E+02 12	2.3E+02 13	2.4E+02 14	2.4E+02 15	0.0E+00
SKC	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	
SKH	2.8E+02	2.5E+02	2.5E+02	2.6E+02	0.0E+00	2.6E+02	2.7E+02	
SKB	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	
SNC	2.5E+02	2.4E+02	2.5E+02	2.6E+02	2.5E+02	0.0E+00	2.6E+02	
SNH	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	7.1E-316	
SNB	2.4E+02	2.5E+02	2.9E+02	2.6E+02	2.6E+02	2.6E+02	2.6E+02	
MKC	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	
MKH	2.5E+02	2.5E+02	2.5E+02	2.5E+02	2.6E+02	2.6E+02	2.6E+02	
MKB	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	
MNC	2.5E+02	2.5E+02	2.5E+02	2.6E+02	2.5E+02	0.0E+00	2.6E+02	
MNH	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	
MNB	2.5E+02	2.5E+02	2.6E+02	2.6E+02	2.6E+02	2.6E+02	2.6E+02	

Table 29: Learned policy for QRF by turn for 15 turn UrbanSim game using DQ-C.

STRATEGY	1	2	3	4	5	6	7	8
SKC	6.3E-316	4.8E-316	2.3E+02	2.3E+02	2.4E+02	2.4E+02	2.4E+02	2.5E+02
SKH	2.4E+02	2.4E+02	2.4E+02	2.4E+02	2.3E+02	2.4E+02	2.4E+02	2.4E+02
SKB	2.4E+02	2.3E+02	2.4E+02	0.0E+00	2.4E+02	2.4E+02	2.4E+02	2.4E+02
SNC	2.3E+02	2.4E+02	2.3E+02	2.4E+02	2.3E+02	2.4E+02	2.4E+02	2.4E+02
SNH	2.6E+02	2.4E+02	2.4E+02	2.3E+02	2.4E+02	2.3E+02	2.4E+02	2.4E+02
SNB	2.6E+02	2.4E+02	2.6E+02	2.4E+02	2.4E+02	2.4E+02	2.4E+02	2.4E+02
MKC	2.3E+02	2.4E+02	2.3E+02	2.6E+02	2.3E+02	2.4E+02	2.4E+02	2.4E+02
MKH	2.4E+02	2.4E+02	2.4E+02	2.4E+02	2.4E+02	0.0E+00	2.4E+02	2.5E+02
MKB	2.4E+02	2.4E+02	2.3E+02	2.4E+02	2.4E+02	2.4E+02	2.4E+02	2.4E+02
MNC	2.4E+02	2.6E+02	2.3E+02	2.4E+02	2.4E+02	2.4E+02	2.4E+02	2.3E+02
MNH	2.4E+02	2.4E+02	2.6E+02	2.3E+02	2.3E+02	2.4E+02	2.4E+02	2.4E+02
MNB	2.3E+02	2.4E+02	2.3E+02	2.3E+02	2.3E+02	2.4E+02	2.4E+02	0.0E+00
	9	10	11	12	13	14	15	
SKC	2.8E+02	2.5E+02	2.5E+02	2.6E+02	2.6E+02	0.0E+00	2.6E+02	
SKH	2.8E+02	2.5E+02	2.5E+02	2.6E+02	0.0E+00	2.6E+02	2.7E+02	
SKB	2.5E+02	0.0E+00	2.6E+02	2.5E+02	2.6E+02	2.6E+02	3.0E+02	
SNC	2.5E+02	2.4E+02	2.5E+02	2.6E+02	2.5E+02	0.0E+00	2.6E+02	
SNH	2.5E+02	2.8E+02	2.5E+02	2.6E+02	2.6E+02	2.6E+02	2.6E+02	
SNB	2.4E+02	2.5E+02	2.9E+02	2.6E+02	2.6E+02	2.6E+02	2.6E+02	
MKC	2.5E+02	2.5E+02	2.5E+02	2.6E+02	2.6E+02	2.6E+02	2.6E+02	
MKH	2.5E+02	2.5E+02	2.5E+02	2.5E+02	2.6E+02	2.6E+02	2.6E+02	
MKB	2.4E+02	2.5E+02	2.5E+02	2.6E+02	2.5E+02	2.6E+02	2.6E+02	
MNC	2.5E+02	2.5E+02	2.5E+02	2.6E+02	2.5E+02	0.0E+00	2.6E+02	
MNH	2.5E+02	2.5E+02	2.5E+02	2.9E+02	2.6E+02	3.0E+02	2.7E+02	
MNB	2.5E+02	2.5E+02	2.6E+02	2.6E+02	2.6E+02	2.6E+02	2.6E+02	

Table 30: Learned policy for H CO b by turn for 15 turn UrbanSim game using DQ-C.

STRATEGY	1	2	3	4	5	6	7	8
SKC	6.3E-316	4.8E-316	2.3E+02	2.3E+02	2.4E+02	2.4E+02	2.7E+02	2.4E+02
SKH	2.4E+02	0.0E+00	2.3E+02	2.4E+02	2.6E+02	2.4E+02	2.4E+02	2.5E+02
SKB	2.4E+02	2.4E+02	0.0E+00	2.4E+02	2.3E+02	2.4E+02	2.4E+02	2.5E+02
SNC	2.4E+02	2.4E+02	2.4E+02	2.3E+02	2.3E+02	2.4E+02	2.4E+02	2.4E+02
SNH	2.4E+02	2.4E+02	2.4E+02	2.3E+02	2.6E+02	2.4E+02	2.4E+02	2.5E+02
SNB	2.4E+02	2.3E+02	2.6E+02	2.4E+02	2.3E+02	2.4E+02	2.4E+02	2.4E+02
MKC	2.4E+02	2.3E+02	2.4E+02	2.4E+02	2.3E+02	2.6E+02	2.4E+02	2.4E+02
MKH	2.3E+02	2.4E+02	2.4E+02	2.4E+02	2.4E+02	2.3E+02	2.4E+02	2.4E+02
MKB	2.3E+02	2.4E+02	2.3E+02	2.4E+02	0.0E+00	2.3E+02	2.4E+02	2.4E+02
MNC	2.4E+02	2.4E+02	2.3E+02	2.3E+02	2.4E+02	2.4E+02	2.4E+02	2.7E+02
MNH	2.4E+02	2.4E+02	2.3E+02	2.6E+02	0.0E+00	2.4E+02	2.4E+02	2.7E+02
MNB	2.3E+02	2.6E+02	2.3E+02	2.4E+02	2.4E+02	2.4E+02	2.4E+02	2.4E+02
	9	10	11	12	13	14	15	
SKC	2.5E+02	0.0E+00	2.5E+02	2.9E+02	2.9E+02	2.6E+02	2.7E+02	
SKH	0.0E+00	2.5E+02	2.5E+02	2.5E+02	2.9E+02	3.0E+02	2.6E+02	
SKB	2.5E+02	2.5E+02	2.6E+02	2.5E+02	2.6E+02	2.6E+02	2.6E+02	
SNC	2.4E+02	2.5E+02	2.5E+02	2.6E+02	2.6E+02	2.7E+02	2.6E+02	
SNH	2.5E+02	2.5E+02	2.5E+02	2.6E+02	2.6E+02	3.0E+02	3.0E+02	
SNB	2.8E+02	2.5E+02	2.9E+02	2.5E+02	2.6E+02	2.6E+02	2.6E+02	
MKC	2.4E+02	2.8E+02	2.6E+02	2.5E+02	2.6E+02	2.6E+02	2.6E+02	
MKH	2.5E+02	2.5E+02	2.5E+02	2.5E+02	2.6E+02	2.6E+02	2.6E+02	
MKB	2.5E+02	2.5E+02	2.5E+02	2.6E+02	2.6E+02	2.7E+02	2.5E+02	
MNC	2.5E+02	2.5E+02	2.5E+02	2.6E+02	2.6E+02	2.7E+02	2.6E+02	
MNH	2.4E+02	2.5E+02	2.5E+02	2.6E+02	2.6E+02	2.6E+02	2.6E+02	
MNB	2.5E+02	2.5E+02	2.5E+02	2.9E+02	2.6E+02	2.6E+02	2.6E+02	

Table 31: Learned policy for Battalion Commander by turn for 15 turn UrbanSim game using Q(λ).

STRATEGY	1	2	3	4	5	6	7	8
SKC	6.3E-316	2.1E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKC	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
	9	10	11	12	13	14	15	
SKC	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKC	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	

Table 32: Learned policy for CA unit by turn for 15 turn UrbanSim game using Q(λ).

STRATEGY	1	2	3	4	5	6	7	8
SKC	7.2E-316	7.7E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKC	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
	9	10	11	12	13	14	15	
SKC	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKC	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	

Table 33: Learned policy for E CO b by turn for 15 turn UrbanSim game using Q(λ).

STRATEGY	1	2	3	4	5	6	7	8
SKC	7.2E-316	7.7E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKC	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
	9	10	11	12	13	14	15	
SKC	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKC	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	

Table 34: Learned policy for E CO a by turn for 15 turn UrbanSim game using Q(λ).

STRATEGY	1	2	3	4	5	6	7	8
SKC	7.2E-316	7.7E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKC	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
	9	10	11	12	13	14	15	
SKC	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKC	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	

Table 35: Learned policy for F CO a by turn for 15 turn UrbanSim game using Q(λ).

STRATEGY	1	2	3	4	5	6	7	8
SKC	6.3E-316	7.7E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKC	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
	9	10	11	12	13	14	15	
SKC	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKC	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	

Table 36: Learned policy for F CO b by turn for 15 turn UrbanSim game using Q(λ).

Strategy	1	2	3	4	5	6	7	8
SKC	6.3E-316	7.7E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKC	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
	9	10	11	12	13	14	15	
SKC	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKC	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	

Table 37: Learned policy for G CO b by turn for 15 turn UrbanSim game using Q(λ).

STRATEGY	1	2	3	4	5	6	7	8
SKC	6.3E-316	7.7E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKC	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
	9	10	11	12	13	14	15	
SKC	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKC	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	

Table 38: Learned policy for G CO a by turn for 15 turn UrbanSim game using Q(λ).

STRATEGY	1	2	3	4	5	6	7	8
SKC	6.3E-316	7.7E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKC	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
	9	10	11	12	13	14	15	
SKC	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKC	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	

Table 39: Learned policy for H CO a by turn for 15 turn UrbanSim game using Q(λ).

STRATEGY	1	2	3	4	5	6	7	8
SKC	6.3E-316	7.7E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
SKH	9.5E+01	9.2E+01	0.0E+00	7.3E+01	4.8E+01	2.8E+01	9.3E+01	3.0E+02
SKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
SNC	3.2E+01	7.1E+01	4.8E+01	8.9E+01	7.3E+01	7.3E+01	5.3E+01	7.0E+01
SNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
SNB	6.0E+01	5.0E+01	2.8E+02	2.8E+02	1.0E+02	7.4E+01	0.0E+00	5.1E+01
MKC	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
MKH	9.3E+01	5.0E+01	1.0E+02	5.1E+01	2.7E+01	2.8E+01	2.6E+01	7.3E+01
MKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
MNC	9.3E+01	7.2E+01	4.9E+01	0.0E+00	6.9E+01	9.5E+01	9.2E+01	9.9E+01
MNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
MNB	8.6E+01	7.1E+01	4.6E+01	0.0E+00	7.4E+01	2.9E+02	7.7E+01	9.5E+01
	9	10	11	12	13	14	15	
SKC	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
SKH	2.7E+01	2.9E+01	5.7E+01	5.7E+01	3.4E+02	3.4E+01	1.1E+02	
SKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
SNC	7.7E+01	1.0E+02	7.8E+01	2.7E+01	0.0E+00	3.1E+01	3.5E+02	
SNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
SNB	7.7E+01	8.1E+01	1.2E+02	3.3E+02	1.0E+02	1.1E+02	8.9E+01	
MKC	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
MKH	7.6E+01	3.0E+01	2.9E+01	8.6E+01	6.0E+01	6.1E+01	1.2E+02	
MKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
MNC	9.2E+01	7.9E+01	5.5E+01	5.7E+01	5.6E+01	8.0E+01	1.3E+02	
MNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
MNB	3.1E+02	3.0E+01	1.2E+02	1.2E+02	8.4E+01	8.1E+01	5.2E+01	

Table 40: Learned policy for QRF by turn for 15 turn UrbanSim game using Q(λ).

Strategy	1	2	3	4	5	6	7	8
SKC	7.2E-316	6.3E-316	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SKH	1.5E-94	3.3E-86	2.2E-57	5.8E+252	3.3E-86	2.2E-57	5.8E+252	3.3E-86
SKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNC	5.8E+252	3.3E-86	2.2E-57	5.8E+252	1.2E-76	7.6E-96	7.3E+199	8.0E-96
SNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNB	1.1E-95	1.3E-71	1.2E+224	1.2E-76	8.1E-96	7.3E+199	3.3E-86	2.2E-57
MKC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKH	8.2E-67	1.2E+224	1.2E-76	8.5E-96	7.3E+199	3.3E-86	2.2E-57	5.8E+252
MKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNC	1.2E+224	1.2E-76	9.0E-96	7.3E+199	3.3E-86	2.2E-57	5.8E+252	1.3E-76
MNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNB	1.2E-76	9.5E-96	7.3E+199	3.3E-86	2.2E-57	5.8E+252	1.3E-76	9.4E-96
	9	10	11	12	13	14	15	
SKC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SKH	2.2E-57	5.8E+252	3.3E-86	2.2E-57	5.8E+252	3.3E-86	2.2E-57	
SKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNC	1.3E-71	1.2E+224	1.2E-76	8.1E-96	7.3E+199	3.3E-86	2.2E-57	
SNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNB	5.8E+252	3.3E-86	2.2E-57	5.8E+252	3.3E-86	2.2E-57	5.8E+252	
MKC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKH	1.3E-76	9.4E-96	1.5E-94	1.3E-259	4.0E+15	8.1E-72	1.3E-259	
MKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNC	9.4E-96	1.5E-94	1.3E-259	4.0E+15	5.3E-67	5.2E-58	2.5E+06	
MNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNB	1.5E-94	1.3E-259	4.0E+15	3.5E-62	5.2E-58	2.5E+06	9.0E-96	

Table 41: Learned policy for H CO b by turn for 15 turn UrbanSim game using Q(λ).

Strategy	1	2	3	4	5	6	7	8
SKC	6.3E-316	7.7E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKC	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
	9	10	11	12	13	14	15	
SKC	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKC	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKB	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNH	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	6.4E-316	
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	

Table 42: Learned policy for Battalion Commander by turn for 15 turn UrbanSim game using SARSA(λ).

STRATEGY	1	2	3	4	5	6	7	8
SKC	6.3E-316	5.1E-316	4.8E-316	4.9E-316	4.9E-316	6.2E-316	4.9E-316	2.0E-316
SKH	-5.3E+10	-5.2E+10	-5.6E+10	-3.3E+10	-3.2E+10	-6.9E+10	-2.7E+10	-7.2E+10
SKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNC	-5.3E+10	-5.1E+10	-5.6E+10	-3.2E+10	-3.2E+10	-6.8E+10	-2.7E+10	-7.2E+10
SNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNB	-5.2E+10	-5.2E+10	-5.6E+10	-3.3E+10	-3.2E+10	-6.8E+10	8.8E+14	3.3E+14
MKC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKH	-5.3E+10	-5.1E+10	-5.5E+10	-3.3E+10	-3.1E+10	-6.9E+10	-2.7E+10	-7.2E+10
MKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNC	6.8E+14	-5.2E+10	-5.6E+10	-3.3E+10	-3.2E+10	-6.9E+10	-2.7E+10	-7.1E+10
MNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNB	-5.3E+10	7.3E+14	-5.6E+10	-3.3E+10	-3.2E+10	-6.8E+10	-2.7E+10	-7.1E+10
	9	10	11	12	13	14	15	
SKC	6.2E-316	6.2E-316	6.2E-316	6.2E-316	6.2E-316	6.2E-316	6.2E-316	
SKH	-2.7E+10	-6.0E+10	-4.1E+10	-7.5E+10	-5.0E+10	-5.2E+10	-4.9E+10	
SKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNC	-2.7E+10	-6.0E+10	-4.1E+10	-7.5E+10	9.4E+14	7.0E+14	7.3E+14	
SNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNB	-2.7E+10	-5.9E+10	-4.1E+10	-7.5E+10	-5.0E+10	-5.2E+10	-4.9E+10	
MKC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKH	-2.7E+10	-6.0E+10	-4.1E+10	-7.6E+10	-5.0E+10	-5.1E+10	-4.9E+10	
MKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNC	-2.7E+10	-6.0E+10	7.8E+14	5.5E+14	-5.1E+10	-5.2E+10	-4.9E+10	
MNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNB	-2.7E+10	-6.0E+10	-4.1E+10	-7.5E+10	-5.0E+10	-5.2E+10	-4.9E+10	

Table 43: Learned policy for CA unit by turn for 15 turn UrbanSim game using SARSA(λ).

STRATEGY	1	2	3	4	5	6	7	8
SKC	6.3E-316	5.1E-316	4.8E-316	4.9E-316	4.9E-316	6.2E-316	4.9E-316	2.0E-316
SKH	-5.4E+12	-5.2E+12	-5.8E+12	-3.0E+12	-2.9E+12	-8.1E+12	-2.4E+12	-8.7E+12
SKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNC	-5.4E+12	-5.2E+12	5.5E+14	-3.0E+12	-2.9E+12	-8.0E+12	-2.4E+12	3.3E+14
SNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNB	-5.5E+12	-5.2E+12	-5.8E+12	-3.0E+12	-2.9E+12	3.5E+14	-2.3E+12	-8.6E+12
MKC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKH	-5.4E+12	-5.2E+12	-5.9E+12	-2.9E+12	-2.9E+12	-8.1E+12	-2.3E+12	-8.6E+12
MKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNC	-5.5E+12	5.7E+14	-5.9E+12	-3.0E+12	5.2E+14	-8.2E+12	-2.4E+12	-8.6E+12
MNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNB	5.5E+14	-5.3E+12	-5.9E+12	-3.0E+12	-2.9E+12	-8.1E+12	-2.3E+12	-8.6E+12
	9	10	11	12	13	14	15	
SKC	6.2E-316	6.2E-316	6.2E-316	6.2E-316	6.2E-316	6.2E-316	6.2E-316	
SKH	-2.4E+12	3.5E+14	-4.0E+12	-8.8E+12	-5.1E+12	-5.3E+12	-5.0E+12	
SKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNC	-2.3E+12	-6.9E+12	-4.0E+12	-8.7E+12	-5.0E+12	-5.2E+12	-5.0E+12	
SNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNB	-2.3E+12	-6.9E+12	-4.0E+12	-8.8E+12	-5.0E+12	-5.3E+12	-5.0E+12	
MKC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKH	-2.3E+12	-6.9E+12	-3.9E+12	-8.8E+12	-5.0E+12	-5.3E+12	-5.0E+12	
MKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNC	-2.3E+12	-6.9E+12	-3.9E+12	-8.8E+12	-5.0E+12	-5.3E+12	-4.9E+12	
MNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNB	-2.4E+12	-6.8E+12	-4.0E+12	-8.7E+12	-5.1E+12	-5.3E+12	-4.9E+12	

Table 44: Learned policy for E CO a by turn for 15 turn UrbanSim game using SARSA(λ).

STRATEGY	1	2	3	4	5	6	7	8
SKC	6.3E-316	5.1E-316	4.8E-316	4.9E-316	4.9E-316	6.2E-316	4.9E-316	2.0E-316
SKH	-2.2E+14	-2.1E+14	-2.7E+14	-2.7E+14	-2.2E+14	-1.7E+14	-2.6E+14	-1.6E+14
SKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNC	-2.8E+14	-2.3E+14	-2.4E+14	-2.3E+14	-1.8E+14	-1.9E+14	-2.2E+14	-1.5E+14
SNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNB	-2.1E+14	-2.8E+14	-2.1E+14	-3.1E+14	-1.9E+14	-1.6E+14	-2.5E+14	-1.7E+14
MKC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKH	-2.3E+14	-2.6E+14	-2.7E+14	-2.9E+14	-2.1E+14	-2.1E+14	-3.2E+14	-2.0E+14
MKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNC	-2.7E+14	-2.9E+14	-2.8E+14	-2.8E+14	-2.5E+14	-1.8E+14	-2.6E+14	-1.9E+14
MNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNB	-2.6E+14	-2.5E+14	-2.3E+14	-2.3E+14	-1.8E+14	-2.0E+14	-3.1E+14	-1.4E+14
	9	10	11	12	13	14	15	
SKC	6.2E-316	6.2E-316	6.2E-316	6.2E-316	6.2E-316	6.2E-316	6.2E-316	
SKH	-2.4E+14	-1.6E+14	-2.6E+14	-2.2E+14	-3.2E+14	-2.5E+14	-2.3E+14	
SKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNC	-2.9E+14	-1.9E+14	-2.1E+14	-2.0E+14	-3.1E+14	-2.3E+14	-3.2E+14	
SNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNB	-2.8E+14	-2.0E+14	-2.7E+14	-2.0E+14	-2.6E+14	-2.8E+14	-3.1E+14	
MKC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKH	-2.5E+14	-1.9E+14	-2.1E+14	-2.3E+14	-2.5E+14	-2.2E+14	-2.4E+14	
MKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNC	-3.1E+14	-1.7E+14	-2.2E+14	-2.4E+14	-2.4E+14	-2.6E+14	-2.5E+14	
MNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNB	-2.7E+14	-1.8E+14	-2.5E+14	-1.7E+14	-2.7E+14	-2.4E+14	-2.6E+14	

Table 45: Learned policy for E CO b by turn for 15 turn UrbanSim game using SARSA(λ).

STRATEGY	1	2	3	4	5	6	7	8
SKC	6.3E-316	5.1E-316	4.8E-316	4.9E-316	4.9E-316	6.2E-316	4.9E-316	2.0E-316
SKH	-2.5E+11	-2.4E+11	-2.6E+11	-1.5E+11	-1.4E+11	-3.3E+11	-1.2E+11	-3.5E+11
SKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNC	-2.5E+11	-2.4E+11	9.9E+14	-1.5E+11	-1.4E+11	-3.3E+11	-1.2E+11	-3.5E+11
SNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNB	-2.4E+11	-2.4E+11	-2.6E+11	1.4E+12	-1.4E+11	-3.4E+11	-1.2E+11	-3.5E+11
MKC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKH	-2.5E+11	-2.4E+11	-2.6E+11	1.1E+15	-1.4E+11	-3.3E+11	-1.2E+11	-3.5E+11
MKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNC	-2.5E+11	1.0E+15	-2.6E+11	-1.5E+11	-1.4E+11	-3.3E+11	-1.2E+11	-3.5E+11
MNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNB	-2.5E+11 9	-2.4E+11 10	-2.6E+11 11	-1.5E+11 12	-1.4E+11 13	-3.3E+11 14	-1.2E+11 15	-3.5E+11
SKC	6.2E-316	6.2E-316	6.2E-316	6.2E-316	6.2E-316	6.2E-316	6.2E-316	
SKH	-1.2E+11	6.2E+14	-1.9E+11	-3.7E+11	1.1E+15	-2.4E+11	-2.3E+11	
SKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNC	-1.2E+11	-2.9E+11	-1.9E+11	-3.6E+11	-2.3E+11	-2.4E+11	-2.3E+11	
SNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNB	-1.2E+11	-2.9E+11	-1.9E+11	-3.7E+11	-2.3E+11	-2.4E+11	-2.3E+11	
MKC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKH	-1.2E+11	-2.9E+11	9.7E+14	-3.7E+11	-2.3E+11	-2.4E+11	-2.3E+11	
MKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNC	1.1E+15	-2.9E+11	-1.9E+11	7.9E+14	-2.3E+11	-2.4E+11	-2.3E+11	
MNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNB	-1.2E+11	-2.9E+11	-1.9E+11	-3.6E+11	-2.3E+11	-2.4E+11	-2.3E+11	

Table 46: Learned policy for F CO b by turn for 15 turn UrbanSim game using SARSA(λ).

STRATEGY	1	2	3	4	5	6	7	8
SKC	6.3E-316	5.1E-316	4.8E-316	4.9E-316	4.9E-316	6.2E-316	4.9E-316	2.0E-316
SKH	-2.6E+13	-2.4E+13	-2.7E+13	-1.3E+13	-1.3E+13	-4.0E+13	2.8E+14	-4.3E+13
SKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNC	-2.6E+13	-2.4E+13	-2.8E+13	-1.3E+13	-1.3E+13	-4.0E+13	-1.0E+13	-4.4E+13
SNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNB	-2.5E+13	-2.4E+13	-2.8E+13	-1.3E+13	-1.3E+13	-4.0E+13	-1.0E+13	-4.4E+13
MKC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKH	-2.6E+13	-2.4E+13	-2.8E+13	-1.3E+13	-1.3E+13	-4.1E+13	-1.0E+13	-4.3E+13
MKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNC	1.8E+14	-2.4E+13	-2.8E+13	-1.3E+13	-1.3E+13	5.3E+13	-1.0E+13	-4.4E+13
MNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNB	-2.6E+13	-2.4E+13	-2.8E+13	2.8E+14	-1.3E+13	-4.0E+13	-1.0E+13	-4.4E+13
	9	10	11	12	13	14	15	
SKC	6.2E-316	6.2E-316	6.2E-316	6.2E-316	6.2E-316	6.2E-316	6.2E-316	
SKH	-1.0E+13	-3.4E+13	-1.8E+13	-4.3E+13	-2.4E+13	-2.5E+13	-2.3E+13	
SKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNC	-1.0E+13	-3.4E+13	-1.8E+13	-4.3E+13	-2.4E+13	-2.5E+13	-2.3E+13	
SNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNB	2.9E+14	-3.4E+13	-1.8E+13	-4.3E+13	-2.3E+13	-2.5E+13	2.1E+14	
MKC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKH	-1.0E+13	-3.4E+13	-1.8E+13	6.2E+13	-2.3E+13	-2.5E+13	-2.3E+13	
MKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNC	-1.0E+13	-3.4E+13	-1.8E+13	-4.3E+13	-2.4E+13	2.0E+14	-2.3E+13	
MNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNB	-1.0E+13	6.7E+13	-1.8E+13	-4.3E+13	-2.4E+13	-2.5E+13	-2.3E+13	

Table 47: Learned policy for F CO a by turn for 15 turn UrbanSim game using SARSA(λ).

STRATEGY	1	2	3	4	5	6	7	8
SKC	6.3E-316	5.1E-316	4.8E-316	4.9E-316	4.9E-316	6.2E-316	4.9E-316	2.0E-316
SKH	-1.2E+14	-7.4E+13	-1.1E+14	-1.2E+14	-8.5E+13	-6.2E+13	-1.4E+14	-4.5E+13
SKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNC	-1.1E+14	-1.1E+14	-9.5E+13	-1.5E+14	-7.8E+13	-5.9E+13	-1.1E+14	-5.0E+13
SNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNB	-8.2E+13	-1.1E+14	-9.9E+13	-1.4E+14	-7.1E+13	-4.3E+13	-8.7E+13	-3.1E+13
MKC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKH	-9.0E+13	-1.2E+14	-7.1E+13	-1.2E+14	-1.1E+14	-3.3E+13	-1.2E+14	-6.1E+13
MKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNC	-1.1E+14	-9.4E+13	-1.2E+14	-1.3E+14	-1.0E+14	-5.6E+13	-1.0E+14	-3.9E+13
MNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNB	-9.5E+13 9	-8.6E+13 10	-8.6E+13 11	-1.0E+14 12	-8.1E+13 13	-6.5E+13 14	-1.5E+14 15	-5.3E+13
SKC	6.2E-316	6.2E-316	6.2E-316	6.2E-316	6.2E-316	6.2E-316	6.2E-316	
SKH	-1.2E+14	-6.3E+13	-1.1E+14	-3.7E+13	-7.9E+13	-1.2E+14	-1.2E+14	
SKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNC	-9.0E+13	-3.8E+13	-7.0E+13	-6.0E+13	-8.8E+13	-1.1E+14	-8.2E+13	
SNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNB	-1.1E+14	-4.4E+13	-1.0E+14	-5.1E+13	-1.3E+14	-9.1E+13	-7.8E+13	
MKC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKH	-1.2E+14	-4.6E+13	-8.4E+13	-4.9E+13	-1.1E+14	-7.5E+13	-7.5E+13	
MKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNC	-1.5E+14	-5.7E+13	-9.2E+13	-7.3E+13	-8.3E+13	-7.2E+13	-8.6E+13	
MNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNB	-1.1E+14	-4.9E+13	-6.7E+13	-6.3E+13	-1.2E+14	-9.6E+13	-1.1E+14	

Table 48: Learned policy for G CO a by turn for 15 turn UrbanSim game using SARSA(λ).

Strategy	1	2	3	4	5	6	7	8
SKC	6.3E-316	5.1E-316	4.8E-316	4.9E-316	4.9E-316	6.2E-316	4.9E-316	2.0E-316
SKH	-1.7E+14	-1.6E+14	-1.8E+14	-1.7E+14	-1.4E+14	-1.0E+14	-1.9E+14	-1.4E+14
SKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNC	-2.1E+14	-2.1E+14	-1.7E+14	-2.4E+14	-1.8E+14	-1.5E+14	-2.2E+14	-1.4E+14
SNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNB	-1.9E+14	-2.0E+14	-1.6E+14	-2.6E+14	-2.0E+14	-1.2E+14	-2.1E+14	-1.1E+14
MKC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKH	-1.5E+14	-1.7E+14	-2.3E+14	-2.1E+14	-1.5E+14	-1.1E+14	-1.7E+14	-1.5E+14
MKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNC	-1.5E+14	-2.4E+14	-2.0E+14	-2.5E+14	-1.5E+14	-1.3E+14	-2.3E+14	-1.2E+14
MNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNB	-1.8E+14 9	-2.2E+14 10	-2.2E+14 11	-2.2E+14 12	-1.4E+14 13	-1.4E+14 14	-2.5E+14 15	-1.3E+14
SKC	6.2E-316	6.2E-316	6.2E-316	6.2E-316	6.2E-316	6.2E-316	6.2E-316	
SKH	-2.0E+14	-1.5E+14	-2.1E+14	-1.6E+14	-1.9E+14	-1.8E+14	-1.8E+14	
SKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNC	-2.6E+14	-1.4E+14	-1.7E+14	-1.2E+14	-2.5E+14	-1.6E+14	-2.3E+14	
SNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNB	-1.7E+14	-1.0E+14	-1.6E+14	-1.7E+14	-2.4E+14	-2.1E+14	-1.9E+14	
MKC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKH	-2.2E+14	-1.4E+14	-2.0E+14	-1.6E+14	-1.8E+14	-2.0E+14	-1.7E+14	
MKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNC	-2.1E+14	-9.8E+13	-1.7E+14	-1.3E+14	-1.7E+14	-1.7E+14	-2.4E+14	
MNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNB	-2.5E+14	-1.3E+14	-1.4E+14	-1.2E+14	-2.6E+14	-1.9E+14	-2.0E+14	

Table 49: Learned policy for G CO b by turn for 15 turn UrbanSim game using SARSA(λ).

Strategy	1	2	3	4	5	6	7	8
SKC	6.3E-316	5.1E-316	4.8E-316	4.9E-316	4.9E-316	6.2E-316	4.9E-316	2.0E-316
SKH	-1.2E+12	-1.1E+12	-1.2E+12	-6.6E+11	-6.5E+11	-1.6E+12	-5.4E+11	-1.7E+12
SKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNC	-1.2E+12	-1.1E+12	-1.2E+12	-6.7E+11	-6.5E+11	-1.6E+12	-5.3E+11	-1.7E+12
SNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNB	-1.2E+12	-1.1E+12	-1.2E+12	-6.5E+11	-6.5E+11	6.1E+14	-5.2E+11	-1.7E+12
MKC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKH	8.7E+14	-1.1E+12	-1.2E+12	-6.7E+11	-6.5E+11	-1.7E+12	-5.3E+11	5.8E+14
MKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNC	-1.2E+12	9.0E+14	-1.2E+12	-6.6E+11	-6.5E+11	-1.6E+12	-5.4E+11	-1.8E+12
MNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNB	-1.2E+12 9	-1.1E+12 10	-1.2E+12 11	-6.6E+11 12	-6.4E+11 13	-1.7E+12 14	-5.3E+11 15	-1.7E+12
SKC	6.2E-316	6.2E-316	6.2E-316	6.2E-316	6.2E-316	6.2E-316	6.2E-316	
SKH	-5.4E+11	-1.4E+12	-8.7E+11	-1.8E+12	-1.1E+12	-1.1E+12	-1.1E+12	
SKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNC	-5.3E+11	-1.4E+12	-8.7E+11	-1.8E+12	-1.1E+12	-1.1E+12	-1.1E+12	
SNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNB	-5.3E+11	-1.4E+12	-8.6E+11	-1.8E+12	-1.1E+12	-1.1E+12	-1.1E+12	
MKC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKH	9.6E+14	-1.4E+12	-8.6E+11	-1.8E+12	-1.1E+12	-1.1E+12	-1.1E+12	
MKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNC	-5.3E+11	-1.4E+12	-8.7E+11	-1.8E+12	9.7E+14	-1.1E+12	-1.1E+12	
MNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNB	-5.4E+11	5.8E+14	-8.7E+11	-1.8E+12	-1.1E+12	-1.1E+12	-1.1E+12	

Table 50: Learned policy for H CO a by turn for 15 turn UrbanSim game using SARSA(λ).

Strategy	1	2	3	4	5	6	7	8
SKC	6.3E-316	5.1E-316	4.8E-316	4.9E-316	4.9E-316	6.2E-316	4.9E-316	2.0E-316
SKH	-1.1E+14	-1.2E+14	-1.1E+14	-1.3E+14	-6.0E+13	-5.8E+13	-1.8E+14	-4.4E+13
SKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNC	-1.1E+14	-1.2E+14	-1.1E+14	-1.3E+14	-6.1E+13	-5.9E+13	-1.7E+14	-4.4E+13
SNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNB	-1.1E+14	-1.2E+14	-1.1E+14	-1.3E+14	-6.0E+13	-5.9E+13	-1.8E+14	-4.4E+13
MKC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKH	-1.1E+14	-1.2E+14	-1.1E+14	-1.3E+14	-2.2E+13	-5.8E+13	-1.6E+14	-4.4E+13
MKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNC	-1.1E+14	-1.2E+14	-1.1E+14	-1.3E+14	-5.9E+13	-5.9E+13	-1.8E+14	-4.4E+13
MNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNB	-1.1E+14	-1.2E+14	-1.1E+14	-1.3E+14	-6.1E+13	-5.9E+13	-1.7E+14	-4.4E+13
	9	10	11	12	13	14	15	
SKC	6.2E-316	6.2E-316	6.2E-316	6.2E-316	6.2E-316	6.2E-316	6.2E-316	
SKH	-1.9E+14	-4.4E+13	-1.4E+14	-8.2E+13	-1.8E+14	-1.1E+14	-1.2E+14	
SKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNC	-1.7E+14	-4.4E+13	-1.6E+14	-8.1E+13	-1.9E+14	-1.0E+14	-1.2E+14	
SNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNB	-1.8E+14	-4.4E+13	-1.6E+14	-8.2E+13	-1.8E+14	-1.1E+14	-1.1E+14	
MKC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKH	-1.8E+14	-4.4E+13	-1.6E+14	-7.1E+13	-2.0E+14	-1.1E+14	-1.2E+14	
MKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNC	-1.7E+14	-4.4E+13	-1.5E+14	-8.2E+13	-1.8E+14	-1.1E+14	-1.2E+14	
MNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNB	-1.7E+14	-4.4E+13	-1.6E+14	-8.2E+13	-2.0E+14	-1.1E+14	-1.2E+14	

Table 51: Learned policy for QRF by turn for 15 turn UrbanSim game using SARSA(λ).

Strategy	1	2	3	4	5	6	7	8
SKC	6.3E-316	5.1E-316	-2.6E+14	-2.4E+14	-2.0E+14	-1.7E+14	-3.0E+14	-1.3E+14
SKH	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316
SKB	-2.4E+14	-2.5E+14	-2.6E+14	-2.4E+14	-1.6E+14	-1.6E+14	-2.9E+14	-1.4E+14
SNC	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316
SNH	-2.3E+14	-2.4E+14	-2.4E+14	-2.8E+14	-2.0E+14	-1.6E+14	-2.4E+14	-1.5E+14
SNB	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316
MKC	-2.1E+14	-2.8E+14	-2.7E+14	-2.3E+14	-1.7E+14	-1.6E+14	-2.4E+14	-1.4E+14
MKH	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316
MKB	-2.4E+14	-2.7E+14	-2.5E+14	-2.6E+14	-1.7E+14	-1.8E+14	-2.5E+14	-1.5E+14
MNC	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316
MNH	-2.1E+14	-2.3E+14	-2.7E+14	-2.7E+14	-1.8E+14	-1.7E+14	-2.7E+14	-1.4E+14
MNB	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316
	9	10	11	12	13	14	15	
SKC	-2.4E+14	-1.4E+14	-2.8E+14	-2.0E+14	-3.2E+14	-2.5E+14	-2.5E+14	
SKH	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	
SKB	-2.5E+14	-1.5E+14	-2.1E+14	-2.2E+14	-2.6E+14	-2.8E+14	-2.7E+14	
SNC	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	
SNH	-2.7E+14	-1.6E+14	-2.6E+14	-1.8E+14	-2.5E+14	-2.7E+14	-2.2E+14	
SNB	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	
MKC	-3.0E+14	-1.4E+14	-2.1E+14	-2.1E+14	-3.2E+14	-2.3E+14	-2.8E+14	
MKH	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	
MKB	-3.0E+14	-1.3E+14	-2.5E+14	-1.9E+14	-2.9E+14	-2.2E+14	-2.2E+14	
MNC	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	
MNH	-2.9E+14	-1.4E+14	-2.3E+14	-1.7E+14	-2.8E+14	-2.4E+14	-2.4E+14	
MNB	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	6.3E-316	

Table 52: Learned policy for H CO b by turn for 15 turn UrbanSim game using SARSA(λ).

Strategy	1	2	3	4	5	6	7	8
SKC	6.3E-316	5.1E-316	4.8E-316	4.9E-316	4.9E-316	6.2E-316	4.9E-316	2.0E-316
SKH	-1.1E+10	-1.9E+13	-7.1E+09	-1.8E+13	-1.6E+13	-6.1E+09	-2.5E+13	-6.1E+09
SKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNC	-1.1E+10	-1.2E+10	-7.3E+09	-1.9E+13	-1.3E+13	-6.1E+09	-4.0E+13	-6.1E+09
SNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNB	-1.1E+10	-1.2E+10	-7.2E+09	-3.9E+13	-1.3E+13	-6.1E+09	-2.7E+13	3.3E+14
MKC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKH	-1.1E+10	-1.2E+10	-7.2E+09	-1.7E+13	-1.5E+13	3.0E+14	-3.4E+13	-6.1E+09
MKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNC	-1.1E+10	-1.2E+10	-7.2E+09	-1.6E+13	-1.2E+13	-6.1E+09	-2.0E+13	-6.2E+09
MNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNB	3.0E+12 9	-1.2E+10 10	-7.2E+09 11	-1.5E+13 12	-1.9E+13 13	-6.1E+09 14	-3.0E+13 15	-6.1E+09
SKC	6.2E-316	6.2E-316	6.2E-316	6.2E-316	6.2E-316	6.2E-316	6.2E-316	
SKH	-2.3E+13	-9.0E+09	-1.8E+13	-1.1E+10	-1.2E+13	-1.1E+10	-1.1E+10	
SKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNC	-3.7E+13	-8.9E+09	-1.6E+10	-1.1E+10	-2.0E+13	-1.1E+10	-1.1E+10	
SNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNB	-2.9E+13	-8.8E+09	-1.5E+13	-1.1E+10	-1.1E+10	-1.1E+10	-1.1E+10	
MKC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKH	-3.9E+13	-8.9E+09	-1.7E+13	-1.1E+10	-1.1E+10	-1.0E+10	-1.1E+10	
MKB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNC	-4.1E+13	-9.0E+09	-1.6E+10	3.1E+14	-1.1E+10	-1.1E+10	-2.0E+13	
MNH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNB	-3.1E+13	-8.9E+09	-1.6E+10	-1.1E+10	-1.1E+10	-1.1E+10	-2.1E+13	

Table 53: Learned policy for Battalion Commander by turn for 15 turn UrbanSim game using DQ-C, Boltzmann.

Strategy	1	2	3	4	5	6	7	8
SKC	9.1E-316	9.5E-316	1.4E-94	1.3E-76	7.6E-96	4.8E+228	5.0E+223	2.2E-57
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SKB	1.3E-76	7.6E-96	4.8E+228	5.0E+223	8.0E-72	9.9E-96	4.0E+252	2.3E+15
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNH	3.4E-53	3.2E+160	1.8E-152	2.6E+180	1.2E-76	1.1E-95	1.2E+214	1.3E+219
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKC	2.6E+180	7.6E-96	1.1E-42	2.9E+161	1.4E+219	8.1E-72	1.3E-76	7.6E-96
MKH	2.5E+02	0.0E+00	0.0E+00	0.0E+00	2.6E+02	0.0E+00	0.0E+00	0.0E+00
MKB	9.3E+252	9.1E+227	1.4E-94	1.3E-76	7.6E-96	4.8E+228	5.0E+223	9.6E-48
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNH	6.0E+175	9.7E-72	4.1E-38	9.3E+252	9.1E+227	1.4E-94	1.0E-42	6.8E-38
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	2.7E+02
	9	10	11	12	13	14	15	
SKC	4.1E-38	9.3E+252	9.1E+227	1.4E-94	1.3E-76	7.6E-96	4.8E+228	
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SKB	1.1E+243	5.2E-67	3.2E-57	4.1E-38	9.3E+252	9.1E+227	1.4E-94	
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	3.1E+02	
SNH	9.7E-72	3.4E-53	3.2E+160	1.8E-152	2.6E+180	1.2E-76	1.1E-95	
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKC	4.8E+228	5.0E+223	1.2E-76	4.1E-38	9.3E+252	9.1E+227	1.4E-94	
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKB	4.1E-38	9.3E+252	9.1E+227	1.4E-94	1.3E-76	7.6E-96	4.8E+228	
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNH	1.2E-95	1.2E+214	1.3E+219	4.2E-62	3.4E-53	3.2E+160	1.8E-152	
MNB	0.0E+00	0.0E+00	2.9E+02	0.0E+00	0.0E+00	0.0E+00	0.0E+00	

Table 54: Learned policy for CA unit by turn for 15 turn UrbanSim game using DQ-C, Boltzmann.

Strategy	1	2	3	4	5	6	7	8
SKC	9.1E-316	9.5E-316	1.4E-94	1.3E-76	7.6E-96	4.8E+228	5.0E+223	2.2E-57
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SKB	1.3E-76	7.6E-96	4.8E+228	5.0E+223	8.0E-72	9.9E-96	4.0E+252	2.3E+15
SNC	0.0E+00	2.5E+02	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNH	3.4E-53	3.2E+160	1.8E-152	2.6E+180	1.2E-76	1.1E-95	1.2E+214	1.3E+219
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	2.7E+02	0.0E+00
MKC	2.6E+180	7.6E-96	1.1E-42	2.9E+161	1.4E+219	8.1E-72	1.3E-76	7.6E-96
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	2.6E+02	0.0E+00	0.0E+00
MKB	9.3E+252	9.1E+227	1.4E-94	1.3E-76	7.6E-96	4.8E+228	5.0E+223	9.6E-48
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNH	6.0E+175	9.7E-72	4.1E-38	9.3E+252	9.1E+227	1.4E-94	1.0E-42	6.8E-38
MNB	0.0E+00	0.0E+00	2.6E+02	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
	9	10	11	12	13	14	15	
SKC	4.1E-38	9.3E+252	9.1E+227	1.4E-94	1.3E-76	7.6E-96	4.8E+228	
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SKB	1.1E+243	5.2E-67	3.2E-57	4.1E-38	9.3E+252	9.1E+227	1.4E-94	
SNC	0.0E+00	0.0E+00	0.0E+00	3.0E+02	0.0E+00	0.0E+00	0.0E+00	
SNH	9.7E-72	3.4E-53	3.2E+160	1.8E-152	2.6E+180	1.2E-76	1.1E-95	
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKC	4.8E+228	5.0E+223	1.2E-76	4.1E-38	9.3E+252	9.1E+227	1.4E-94	
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKB	4.1E-38	9.3E+252	9.1E+227	1.4E-94	1.3E-76	7.6E-96	4.8E+228	
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNH	1.2E-95	1.2E+214	1.3E+219	4.2E-62	3.4E-53	3.2E+160	1.8E-152	
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	

Table 55: Learned policy for E CO a by turn for 15 turn UrbanSim game using DQ-C, Boltzmann.

Strategy	1	2	3	4	5	6	7	8
SKC	9.1E-316	9.5E-316	1.4E-94	1.3E-76	7.6E-96	4.8E+228	5.0E+223	2.2E-57
SKH	0.0E+00	2.5E+02	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SKB	1.3E-76	7.6E-96	4.8E+228	5.0E+223	8.0E-72	9.9E-96	4.0E+252	2.3E+15
SNC	0.0E+00	0.0E+00	0.0E+00	2.6E+02	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNH	3.4E-53	3.2E+160	1.8E-152	2.6E+180	1.2E-76	1.1E-95	1.2E+214	1.3E+219
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKC	2.6E+180	7.6E-96	1.1E-42	2.9E+161	1.4E+219	8.1E-72	1.3E-76	7.6E-96
MKH	0.0E+00	0.0E+00	2.6E+02	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKB	9.3E+252	9.1E+227	1.4E-94	1.3E-76	7.6E-96	4.8E+228	5.0E+223	9.6E-48
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	2.7E+02	0.0E+00
MNH	6.0E+175	9.7E-72	4.1E-38	9.3E+252	9.1E+227	1.4E-94	1.0E-42	6.8E-38
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
	9	10	11	12	13	14	15	
SKC	4.1E-38	9.3E+252	9.1E+227	1.4E-94	1.3E-76	7.6E-96	4.8E+228	
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SKB	1.1E+243	5.2E-67	3.2E-57	4.1E-38	9.3E+252	9.1E+227	1.4E-94	
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	3.0E+02	3.0E+02	0.0E+00	
SNH	9.7E-72	3.4E-53	3.2E+160	1.8E-152	2.6E+180	1.2E-76	1.1E-95	
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKC	4.8E+228	5.0E+223	1.2E-76	4.1E-38	9.3E+252	9.1E+227	1.4E-94	
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKB	4.1E-38	9.3E+252	9.1E+227	1.4E-94	1.3E-76	7.6E-96	4.8E+228	
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNH	1.2E-95	1.2E+214	1.3E+219	4.2E-62	3.4E-53	3.2E+160	1.8E-152	
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	

Table 56: Learned policy for E CO b by turn for 15 turn UrbanSim game using DQ-C, Boltzmann.

Strategy	1	2	3	4	5	6	7	8
SKC	9.1E-316	9.5E-316	1.4E-94	1.3E-76	7.6E-96	4.8E+228	5.0E+223	2.2E-57
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SKB	1.3E-76	7.6E-96	4.8E+228	5.0E+223	8.0E-72	9.9E-96	4.0E+252	2.3E+15
SNC	0.0E+00	0.0E+00	0.0E+00	2.6E+02	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNH	3.4E-53	3.2E+160	1.8E-152	2.6E+180	1.2E-76	1.1E-95	1.2E+214	1.3E+219
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	2.7E+02	0.0E+00
MKC	2.6E+180	7.6E-96	1.1E-42	2.9E+161	1.4E+219	8.1E-72	1.3E-76	7.6E-96
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	2.6E+02	0.0E+00	0.0E+00	0.0E+00
MKB	9.3E+252	9.1E+227	1.4E-94	1.3E-76	7.6E-96	4.8E+228	5.0E+223	9.6E-48
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNH	6.0E+175	9.7E-72	4.1E-38	9.3E+252	9.1E+227	1.4E-94	1.0E-42	6.8E-38
MNB	2.5E+02	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
	9	10	11	12	13	14	15	
SKC	4.1E-38	9.3E+252	9.1E+227	1.4E-94	1.3E-76	7.6E-96	4.8E+228	
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SKB	1.1E+243	5.2E-67	3.2E-57	4.1E-38	9.3E+252	9.1E+227	1.4E-94	
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	3.1E+02	
SNH	9.7E-72	3.4E-53	3.2E+160	1.8E-152	2.6E+180	1.2E-76	1.1E-95	
SNB	0.0E+00	2.8E+02	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKC	4.8E+228	5.0E+223	1.2E-76	4.1E-38	9.3E+252	9.1E+227	1.4E-94	
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKB	4.1E-38	9.3E+252	9.1E+227	1.4E-94	1.3E-76	7.6E-96	4.8E+228	
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	3.0E+02	0.0E+00	
MNH	1.2E-95	1.2E+214	1.3E+219	4.2E-62	3.4E-53	3.2E+160	1.8E-152	
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	

Table 57: Learned policy for F CO a by turn for 15 turn UrbanSim game using DQ-C, Boltzmann.

Strategy	1	2	3	4	5	6	7	8
SKC	6.4E-316	9.5E-316	1.4E-94	1.3E-76	7.6E-96	4.8E+228	5.0E+223	2.2E-57
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SKB	1.3E-76	7.6E-96	4.8E+228	5.0E+223	8.0E-72	9.9E-96	4.0E+252	2.3E+15
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNH	3.4E-53	3.2E+160	1.8E-152	2.6E+180	1.2E-76	1.1E-95	1.2E+214	1.3E+219
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	2.6E+02	0.0E+00	0.0E+00	0.0E+00
MKC	2.6E+180	7.6E-96	1.1E-42	2.9E+161	1.4E+219	8.1E-72	1.3E-76	7.6E-96
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	2.7E+02
MKB	9.3E+252	9.1E+227	1.4E-94	1.3E-76	7.6E-96	4.8E+228	5.0E+223	9.6E-48
MNC	0.0E+00	2.5E+02	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNH	6.0E+175	9.7E-72	4.1E-38	9.3E+252	9.1E+227	1.4E-94	1.0E-42	6.8E-38
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
	9	10	11	12	13	14	15	
SKC	4.1E-38	9.3E+252	9.1E+227	1.4E-94	1.3E-76	7.6E-96	4.8E+228	
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SKB	1.1E+243	5.2E-67	3.2E-57	4.1E-38	9.3E+252	9.1E+227	1.4E-94	
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	3.0E+02	0.0E+00	0.0E+00	
SNH	9.7E-72	3.4E-53	3.2E+160	1.8E-152	2.6E+180	1.2E-76	1.1E-95	
SNB	2.8E+02	2.8E+02	2.9E+02	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKC	4.8E+228	5.0E+223	1.2E-76	4.1E-38	9.3E+252	9.1E+227	1.4E-94	
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKB	4.1E-38	9.3E+252	9.1E+227	1.4E-94	1.3E-76	7.6E-96	4.8E+228	
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNH	1.2E-95	1.2E+214	1.3E+219	4.2E-62	3.4E-53	3.2E+160	1.8E-152	
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	3.1E+02	

Table 58: Learned policy for F CO b by turn for 15 turn UrbanSim game using DQ-C, Boltzmann.

Strategy	1	2	3	4	5	6	7	8
SKC	6.4E-316	9.5E-316	1.4E-94	1.3E-76	7.6E-96	4.8E+228	5.0E+223	2.2E-57
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SKB	1.3E-76	7.6E-96	4.8E+228	5.0E+223	8.0E-72	9.9E-96	4.0E+252	2.3E+15
SNC	0.0E+00	0.0E+00	0.0E+00	2.6E+02	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNH	3.4E-53	3.2E+160	1.8E-152	2.6E+180	1.2E-76	1.1E-95	1.2E+214	1.3E+219
SNB	0.0E+00	0.0E+00	2.6E+02	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKC	2.6E+180	7.6E-96	1.1E-42	2.9E+161	1.4E+219	8.1E-72	1.3E-76	7.6E-96
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	2.6E+02	0.0E+00	2.7E+02	2.7E+02
MKB	9.3E+252	9.1E+227	1.4E-94	1.3E-76	7.6E-96	4.8E+228	5.0E+223	9.6E-48
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNH	6.0E+175	9.7E-72	4.1E-38	9.3E+252	9.1E+227	1.4E-94	1.0E-42	6.8E-38
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
	9	10	11	12	13	14	15	
SKC	4.1E-38	9.3E+252	9.1E+227	1.4E-94	1.3E-76	7.6E-96	4.8E+228	
SKH	2.8E+02	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SKB	1.1E+243	5.2E-67	3.2E-57	4.1E-38	9.3E+252	9.1E+227	1.4E-94	
SNC	0.0E+00	0.0E+00	0.0E+00	3.0E+02	0.0E+00	0.0E+00	0.0E+00	
SNH	9.7E-72	3.4E-53	3.2E+160	1.8E-152	2.6E+180	1.2E-76	1.1E-95	
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKC	4.8E+228	5.0E+223	1.2E-76	4.1E-38	9.3E+252	9.1E+227	1.4E-94	
MKH	0.0E+00	2.8E+02	0.0E+00	0.0E+00	0.0E+00	0.0E+00	3.1E+02	
MKB	4.1E-38	9.3E+252	9.1E+227	1.4E-94	1.3E-76	7.6E-96	4.8E+228	
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNH	1.2E-95	1.2E+214	1.3E+219	4.2E-62	3.4E-53	3.2E+160	1.8E-152	
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	

Table 59: Learned policy for G CO a by turn for 15 turn UrbanSim game using DQ-C, Boltzmann.

Strategy	1	2	3	4	5	6	7	8
SKC	6.4E-316	9.5E-316	1.4E-94	1.3E-76	7.6E-96	4.8E+228	5.0E+223	2.2E-57
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	2.7E+02	0.0E+00
SKB	1.3E-76	7.6E-96	4.8E+228	5.0E+223	8.0E-72	9.9E-96	4.0E+252	2.3E+15
SNC	0.0E+00	0.0E+00	2.6E+02	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNH	3.4E-53	3.2E+160	1.8E-152	2.6E+180	1.2E-76	1.1E-95	1.2E+214	1.3E+219
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKC	2.6E+180	7.6E-96	1.1E-42	2.9E+161	1.4E+219	8.1E-72	1.3E-76	7.6E-96
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKB	9.3E+252	9.1E+227	1.4E-94	1.3E-76	7.6E-96	4.8E+228	5.0E+223	9.6E-48
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	2.6E+02	0.0E+00	0.0E+00
MNH	6.0E+175	9.7E-72	4.1E-38	9.3E+252	9.1E+227	1.4E-94	1.0E-42	6.8E-38
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	2.7E+02
	9	10	11	12	13	14	15	
SKC	4.1E-38	9.3E+252	9.1E+227	1.4E-94	1.3E-76	7.6E-96	4.8E+228	
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SKB	1.1E+243	5.2E-67	3.2E-57	4.1E-38	9.3E+252	9.1E+227	1.4E-94	
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	3.1E+02	
SNH	9.7E-72	3.4E-53	3.2E+160	1.8E-152	2.6E+180	1.2E-76	1.1E-95	
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKC	4.8E+228	5.0E+223	1.2E-76	4.1E-38	9.3E+252	9.1E+227	1.4E-94	
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKB	4.1E-38	9.3E+252	9.1E+227	1.4E-94	1.3E-76	7.6E-96	4.8E+228	
MNC	0.0E+00	0.0E+00	0.0E+00	3.0E+02	0.0E+00	0.0E+00	0.0E+00	
MNH	1.2E-95	1.2E+214	1.3E+219	4.2E-62	3.4E-53	3.2E+160	1.8E-152	
MNB	0.0E+00	2.8E+02	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	

Table 60: Learned policy for G CO b by turn for 15 turn UrbanSim game using DQ-C, Boltzmann.

Strategy	1	2	3	4	5	6	7	8
SKC	6.4E-316	9.5E-316	1.4E-94	1.3E-76	7.6E-96	4.8E+228	5.0E+223	2.2E-57
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SKB	1.3E-76	7.6E-96	4.8E+228	5.0E+223	8.0E-72	9.9E-96	4.0E+252	2.3E+15
SNC	0.0E+00	2.5E+02	0.0E+00	2.6E+02	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNH	3.4E-53	3.2E+160	1.8E-152	2.6E+180	1.2E-76	1.1E-95	1.2E+214	1.3E+219
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKC	2.6E+180	7.6E-96	1.1E-42	2.9E+161	1.4E+219	8.1E-72	1.3E-76	7.6E-96
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKB	9.3E+252	9.1E+227	1.4E-94	1.3E-76	7.6E-96	4.8E+228	5.0E+223	9.6E-48
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNH	6.0E+175	9.7E-72	4.1E-38	9.3E+252	9.1E+227	1.4E-94	1.0E-42	6.8E-38
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
	9	10	11	12	13	14	15	
SKC	4.1E-38	9.3E+252	9.1E+227	1.4E-94	1.3E-76	7.6E-96	4.8E+228	
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SKB	1.1E+243	5.2E-67	3.2E-57	4.1E-38	9.3E+252	9.1E+227	1.4E-94	
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNH	9.7E-72	3.4E-53	3.2E+160	1.8E-152	2.6E+180	1.2E-76	1.1E-95	
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKC	4.8E+228	5.0E+223	1.2E-76	4.1E-38	9.3E+252	9.1E+227	1.4E-94	
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKB	4.1E-38	9.3E+252	9.1E+227	1.4E-94	1.3E-76	7.6E-96	4.8E+228	
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNH	1.2E-95	1.2E+214	1.3E+219	4.2E-62	3.4E-53	3.2E+160	1.8E-152	
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	

Table 61: Learned policy for H CO a by turn for 15 turn UrbanSim game using DQ-C, Boltzmann.

Strategy	1	2	3	4	5	6	7	8
SKC	6.4E-316	9.5E-316	1.4E-94	1.3E-76	7.6E-96	4.8E+228	5.0E+223	2.2E-57
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SKB	1.3E-76	7.6E-96	4.8E+228	5.0E+223	8.0E-72	9.9E-96	4.0E+252	2.3E+15
SNC	0.0E+00	2.5E+02	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNH	3.4E-53	3.2E+160	1.8E-152	2.6E+180	1.2E-76	1.1E-95	1.2E+214	1.3E+219
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKC	2.6E+180	7.6E-96	1.1E-42	2.9E+161	1.4E+219	8.1E-72	1.3E-76	7.6E-96
MKH	2.5E+02	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKB	9.3E+252	9.1E+227	1.4E-94	1.3E-76	7.6E-96	4.8E+228	5.0E+223	9.6E-48
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNH	6.0E+175	9.7E-72	4.1E-38	9.3E+252	9.1E+227	1.4E-94	1.0E-42	6.8E-38
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	2.6E+02	0.0E+00	0.0E+00	2.7E+02
	9	10	11	12	13	14	15	
SKC	4.1E-38	9.3E+252	9.1E+227	1.4E-94	1.3E-76	7.6E-96	4.8E+228	
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SKB	1.1E+243	5.2E-67	3.2E-57	4.1E-38	9.3E+252	9.1E+227	1.4E-94	
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNH	9.7E-72	3.4E-53	3.2E+160	1.8E-152	2.6E+180	1.2E-76	1.1E-95	
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKC	4.8E+228	5.0E+223	1.2E-76	4.1E-38	9.3E+252	9.1E+227	1.4E-94	
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKB	4.1E-38	9.3E+252	9.1E+227	1.4E-94	1.3E-76	7.6E-96	4.8E+228	
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNH	1.2E-95	1.2E+214	1.3E+219	4.2E-62	3.4E-53	3.2E+160	1.8E-152	
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	

Table 62: Learned policy for H CO b by turn for 15 turn UrbanSim game using DQ-C, Boltzmann.

Strategy	1	2	3	4	5	6	7	8
SKC	6.4E-316	9.5E-316	1.4E-94	1.3E-76	7.6E-96	4.8E+228	5.0E+223	2.2E-57
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	2.6E+02	0.0E+00	0.0E+00	0.0E+00
SKB	1.3E-76	7.6E-96	4.8E+228	5.0E+223	8.0E-72	9.9E-96	4.0E+252	2.3E+15
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNH	3.4E-53	3.2E+160	1.8E-152	2.6E+180	1.2E-76	1.1E-95	1.2E+214	1.3E+219
SNB	2.5E+02	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKC	2.6E+180	7.6E-96	1.1E-42	2.9E+161	1.4E+219	8.1E-72	1.3E-76	7.6E-96
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKB	9.3E+252	9.1E+227	1.4E-94	1.3E-76	7.6E-96	4.8E+228	5.0E+223	9.6E-48
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	2.7E+02
MNH	6.0E+175	9.7E-72	4.1E-38	9.3E+252	9.1E+227	1.4E-94	1.0E-42	6.8E-38
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	2.7E+02	0.0E+00
	9	10	11	12	13	14	15	
SKC	4.1E-38	9.3E+252	9.1E+227	1.4E-94	1.3E-76	7.6E-96	4.8E+228	
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	3.1E+02	
SKB	1.1E+243	5.2E-67	3.2E-57	4.1E-38	9.3E+252	9.1E+227	1.4E-94	
SNC	0.0E+00	0.0E+00	2.9E+02	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNH	9.7E-72	3.4E-53	3.2E+160	1.8E-152	2.6E+180	1.2E-76	1.1E-95	
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	3.0E+02	0.0E+00	0.0E+00	
MKC	4.8E+228	5.0E+223	1.2E-76	4.1E-38	9.3E+252	9.1E+227	1.4E-94	
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKB	4.1E-38	9.3E+252	9.1E+227	1.4E-94	1.3E-76	7.6E-96	4.8E+228	
MNC	2.8E+02	0.0E+00	0.0E+00	0.0E+00	0.0E+00	3.0E+02	0.0E+00	
MNH	1.2E-95	1.2E+214	1.3E+219	4.2E-62	3.4E-53	3.2E+160	1.8E-152	
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	

Table 63: Learned policy for QRF by turn for 15 turn UrbanSim game using DQ-C, Boltzmann.

Strategy	1	2	3	4	5	6	7	8
SKC	6.4E-316	9.5E-316	1.4E-94	1.3E-76	7.6E-96	4.8E+228	5.0E+223	2.2E-57
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SKB	1.3E-76	7.6E-96	4.8E+228	5.0E+223	8.0E-72	9.9E-96	4.0E+252	2.3E+15
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNH	3.4E-53	3.2E+160	1.8E-152	2.6E+180	1.2E-76	1.1E-95	1.2E+214	1.3E+219
SNB	2.5E+02	0.0E+00	2.6E+02	0.0E+00	2.6E+02	0.0E+00	2.7E+02	0.0E+00
MKC	2.6E+180	7.6E-96	1.1E-42	2.9E+161	1.4E+219	8.1E-72	1.3E-76	7.6E-96
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKB	9.3E+252	9.1E+227	1.4E-94	1.3E-76	7.6E-96	4.8E+228	5.0E+223	9.6E-48
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	2.6E+02	0.0E+00	2.7E+02
MNH	6.0E+175	9.7E-72	4.1E-38	9.3E+252	9.1E+227	1.4E-94	1.0E-42	6.8E-38
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
	9	10	11	12	13	14	15	
SKC	4.1E-38	9.3E+252	9.1E+227	1.4E-94	1.3E-76	7.6E-96	4.8E+228	
SKH	2.8E+02	0.0E+00	0.0E+00	3.0E+02	0.0E+00	0.0E+00	0.0E+00	
SKB	1.1E+243	5.2E-67	3.2E-57	4.1E-38	9.3E+252	9.1E+227	1.4E-94	
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	3.0E+02	0.0E+00	
SNH	9.7E-72	3.4E-53	3.2E+160	1.8E-152	2.6E+180	1.2E-76	1.1E-95	
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKC	4.8E+228	5.0E+223	1.2E-76	4.1E-38	9.3E+252	9.1E+227	1.4E-94	
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	3.0E+02	0.0E+00	0.0E+00	
MKB	4.1E-38	9.3E+252	9.1E+227	1.4E-94	1.3E-76	7.6E-96	4.8E+228	
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNH	1.2E-95	1.2E+214	1.3E+219	4.2E-62	3.4E-53	3.2E+160	1.8E-152	
MNB	0.0E+00	0.0E+00	2.9E+02	0.0E+00	0.0E+00	0.0E+00	0.0E+00	

Table 64: Learned policy for Battalion Commander by turn for 15 turn UrbanSim game using Q(λ), Boltzmann.

Strategy	1	2	3	4	5	6	7	8
SKC	6.3E-316	4.7E-316	1.2E+166	2.4E+232	1.4E-303	2.0E-316	6.0E-154	1.1E-152
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SKB	1.8E-152	1.7E+214	2.5E-154	1.4E+219	1.9E+219	4.7E+252	7.7E+170	7.3E+223
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNH	5.3E+228	3.0E+222	1.2E-259	2.1E-312	1.1E-311	2.7E-308	2.0E-309	2.1E-301
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKC	4.6E+228	1.3E-152	2.0E+161	3.3E+257	2.5E+198	1.7E-152	1.2E+214	4.0E+252
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKB	4.5E+257	5.8E+180	5.8E+252	1.8E+190	1.3E+213	5.9E+199	2.5E-154	1.7E+214
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNH	1.8E-313	1.4E-308	9.9E+86	7.2E-311	7.1E-235	3.5E-313	2.5E-308	9.9E+86
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
Strategy	9	10	11	12	13	14	15	
SKC	3.0E+180	5.0E+175	2.2E+161	9.8E+199	4.7E+257	1.9E+227	8.0E+165	
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SKB	4.7E+164	2.1E+214	3.0E-85	6.0E-154	2.0E+267	1.7E+243	2.3E+243	
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNH	1.6E-309	6.7E-314	6.2E-309	1.8E+175	4.6E-314	9.9E-307	5.1E-116	
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKC	3.0E-85	1.7E+262	3.1E+169	1.9E+214	1.8E+185	5.8E+252	2.0E+161	
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKB	1.8E-152	6.0E+247	5.8E+180	5.3E-85	3.4E-309	6.0E+197	3.0E-311	
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNH	1.2E-310	2.4E-196	5.2E-313	3.7E-308	9.9E+86	1.6E-310	8.2E-158	
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	

Table 65: Learned policy for CA unit by turn for 15 turn UrbanSim game using Q(λ), Boltzmann.

Strategy	1	2	3	4	5	6	7	8
SKC	6.3E-316	4.7E-316	1.2E+166	2.4E+232	1.4E-303	2.0E-316	6.0E-154	1.1E-152
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SKB	1.8E-152	1.7E+214	2.5E-154	1.4E+219	1.9E+219	4.7E+252	7.7E+170	7.3E+223
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNH	5.3E+228	3.0E+222	1.2E-259	2.1E-312	1.1E-311	2.7E-308	2.0E-309	2.1E-301
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKC	4.6E+228	1.3E-152	2.0E+161	3.3E+257	2.5E+198	1.7E-152	1.2E+214	4.0E+252
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKB	4.5E+257	5.8E+180	5.8E+252	1.8E+190	1.3E+213	5.9E+199	2.5E-154	1.7E+214
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNH	1.8E-313	1.4E-308	9.9E+86	7.2E-311	7.1E-235	3.5E-313	2.5E-308	9.9E+86
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
Strategy	9	10	11	12	13	14	15	
SKC	3.0E+180	5.0E+175	2.2E+161	9.8E+199	4.7E+257	1.9E+227	8.0E+165	
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SKB	4.7E+164	2.1E+214	3.0E-85	6.0E-154	2.0E+267	1.7E+243	2.3E+243	
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNH	1.6E-309	6.7E-314	6.2E-309	1.8E+175	4.6E-314	9.9E-307	5.1E-116	
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKC	3.0E-85	1.7E+262	3.1E+169	1.9E+214	1.8E+185	5.8E+252	2.0E+161	
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKB	1.8E-152	6.0E+247	5.8E+180	5.3E-85	3.4E-309	6.0E+197	3.0E-311	
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNH	1.2E-310	2.4E-196	5.2E-313	3.7E-308	9.9E+86	1.6E-310	8.2E-158	
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	

Table 66: Learned policy for E CO a by turn for 15 turn UrbanSim game using Q(λ), Boltzmann.

Strategy	1	2	3	4	5	6	7	8
SKC	6.3E-316	4.7E-316	1.2E+166	2.4E+232	1.4E-303	2.0E-316	6.0E-154	1.1E-152
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SKB	1.8E-152	1.7E+214	2.5E-154	1.4E+219	1.9E+219	4.7E+252	7.7E+170	7.3E+223
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNH	5.3E+228	3.0E+222	1.2E-259	2.1E-312	1.1E-311	2.7E-308	2.0E-309	2.1E-301
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKC	4.6E+228	1.3E-152	2.0E+161	3.3E+257	2.5E+198	1.7E-152	1.2E+214	4.0E+252
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKB	4.5E+257	5.8E+180	5.8E+252	1.8E+190	1.3E+213	5.9E+199	2.5E-154	1.7E+214
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNH	1.8E-313	1.4E-308	9.9E+86	7.2E-311	7.1E-235	3.5E-313	2.5E-308	9.9E+86
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
Strategy	9	10	11	12	13	14	15	
SKC	3.0E+180	5.0E+175	2.2E+161	9.8E+199	4.7E+257	1.9E+227	8.0E+165	
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SKB	4.7E+164	2.1E+214	3.0E-85	6.0E-154	2.0E+267	1.7E+243	2.3E+243	
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNH	1.6E-309	6.7E-314	6.2E-309	1.8E+175	4.6E-314	9.9E-307	5.1E-116	
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKC	3.0E-85	1.7E+262	3.1E+169	1.9E+214	1.8E+185	5.8E+252	2.0E+161	
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKB	1.8E-152	6.0E+247	5.8E+180	5.3E-85	3.4E-309	6.0E+197	3.0E-311	
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNH	1.2E-310	2.4E-196	5.2E-313	3.7E-308	9.9E+86	1.6E-310	8.2E-158	
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	

Table 67: Learned policy for F CO a by turn for 15 turn UrbanSim game using Q(λ), Boltzmann.

Strategy	1	2	3	4	5	6	7	8
SKC	6.3E-316	4.7E-316	1.2E+166	2.4E+232	1.4E-303	2.0E-316	6.0E-154	1.1E-152
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SKB	1.8E-152	1.7E+214	2.5E-154	1.4E+219	1.9E+219	4.7E+252	7.7E+170	7.3E+223
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNH	5.3E+228	3.0E+222	1.2E-259	2.1E-312	1.1E-311	2.7E-308	2.0E-309	2.1E-301
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKC	4.6E+228	1.3E-152	2.0E+161	3.3E+257	2.5E+198	1.7E-152	1.2E+214	4.0E+252
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKB	4.5E+257	5.8E+180	5.8E+252	1.8E+190	1.3E+213	5.9E+199	2.5E-154	1.7E+214
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNH	1.8E-313	1.4E-308	9.9E+86	7.2E-311	7.1E-235	3.5E-313	2.5E-308	9.9E+86
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
Strategy	9	10	11	12	13	14	15	
SKC	3.0E+180	5.0E+175	2.2E+161	9.8E+199	4.7E+257	1.9E+227	8.0E+165	
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SKB	4.7E+164	2.1E+214	3.0E-85	6.0E-154	2.0E+267	1.7E+243	2.3E+243	
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNH	1.6E-309	6.7E-314	6.2E-309	1.8E+175	4.6E-314	9.9E-307	5.1E-116	
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKC	3.0E-85	1.7E+262	3.1E+169	1.9E+214	1.8E+185	5.8E+252	2.0E+161	
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKB	1.8E-152	6.0E+247	5.8E+180	5.3E-85	3.4E-309	6.0E+197	3.0E-311	
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNH	1.2E-310	2.4E-196	5.2E-313	3.7E-308	9.9E+86	1.6E-310	8.2E-158	
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	

Table 68: Learned policy for E CO b by turn for 15 turn UrbanSim game using Q(λ), Boltzmann.

Strategy	1	2	3	4	5	6	7	8
SKC	6.3E-316	6.4E-316	1.2E+166	2.4E+232	1.4E-303	2.0E-316	6.0E-154	1.1E-152
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SKB	1.8E-152	1.7E+214	2.5E-154	1.4E+219	1.9E+219	4.7E+252	7.7E+170	7.3E+223
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNH	5.3E+228	3.0E+222	1.2E-259	2.1E-312	1.1E-311	2.7E-308	2.0E-309	2.1E-301
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKC	4.6E+228	1.3E-152	2.0E+161	3.3E+257	2.5E+198	1.7E-152	1.2E+214	4.0E+252
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKB	4.5E+257	5.8E+180	5.8E+252	1.8E+190	1.3E+213	5.9E+199	2.5E-154	1.7E+214
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNH	1.8E-313	1.4E-308	9.9E+86	7.2E-311	7.1E-235	3.5E-313	2.5E-308	9.9E+86
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
Strategy	9	10	11	12	13	14	15	
SKC	3.0E+180	5.0E+175	2.2E+161	9.8E+199	4.7E+257	1.9E+227	8.0E+165	
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SKB	4.7E+164	2.1E+214	3.0E-85	6.0E-154	2.0E+267	1.7E+243	2.3E+243	
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNH	1.6E-309	6.7E-314	6.2E-309	1.8E+175	4.6E-314	9.9E-307	5.1E-116	
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKC	3.0E-85	1.7E+262	3.1E+169	1.9E+214	1.8E+185	5.8E+252	2.0E+161	
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKB	1.8E-152	6.0E+247	5.8E+180	5.3E-85	3.4E-309	6.0E+197	3.0E-311	
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNH	1.2E-310	2.4E-196	5.2E-313	3.7E-308	9.9E+86	1.6E-310	8.2E-158	
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	

Table 69: Learned policy for F CO b by turn for 15 turn UrbanSim game using Q(λ), Boltzmann.

Strategy	1	2	3	4	5	6	7	8
SKC	6.3E-316	4.7E-316	1.2E+166	2.4E+232	1.4E-303	2.0E-316	6.0E-154	1.1E-152
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SKB	1.8E-152	1.7E+214	2.5E-154	1.4E+219	1.9E+219	4.7E+252	7.7E+170	7.3E+223
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNH	5.3E+228	3.0E+222	1.2E-259	2.1E-312	1.1E-311	2.7E-308	2.0E-309	2.1E-301
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKC	4.6E+228	1.3E-152	2.0E+161	3.3E+257	2.5E+198	1.7E-152	1.2E+214	4.0E+252
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKB	4.5E+257	5.8E+180	5.8E+252	1.8E+190	1.3E+213	5.9E+199	2.5E-154	1.7E+214
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNH	1.8E-313	1.4E-308	9.9E+86	7.2E-311	7.1E-235	3.5E-313	2.5E-308	9.9E+86
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
Strategy	9	10	11	12	13	14	15	
SKC	3.0E+180	5.0E+175	2.2E+161	9.8E+199	4.7E+257	1.9E+227	8.0E+165	
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SKB	4.7E+164	2.1E+214	3.0E-85	6.0E-154	2.0E+267	1.7E+243	2.3E+243	
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNH	1.6E-309	6.7E-314	6.2E-309	1.8E+175	4.6E-314	9.9E-307	5.1E-116	
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKC	3.0E-85	1.7E+262	3.1E+169	1.9E+214	1.8E+185	5.8E+252	2.0E+161	
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKB	1.8E-152	6.0E+247	5.8E+180	5.3E-85	3.4E-309	6.0E+197	3.0E-311	
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNH	1.2E-310	2.4E-196	5.2E-313	3.7E-308	9.9E+86	1.6E-310	8.2E-158	
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	

Table 70: Learned policy for G CO b by turn for 15 turn UrbanSim game using Q(λ), Boltzmann.

Strategy	1	2	3	4	5	6	7	8
SKC	6.3E-316	4.7E-316	1.2E+166	2.4E+232	1.4E-303	2.0E-316	6.0E-154	1.1E-152
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SKB	1.8E-152	1.7E+214	2.5E-154	1.4E+219	1.9E+219	4.7E+252	7.7E+170	7.3E+223
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNH	5.3E+228	3.0E+222	1.2E-259	2.1E-312	1.1E-311	2.7E-308	2.0E-309	2.1E-301
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKC	4.6E+228	1.3E-152	2.0E+161	3.3E+257	2.5E+198	1.7E-152	1.2E+214	4.0E+252
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKB	4.5E+257	5.8E+180	5.8E+252	1.8E+190	1.3E+213	5.9E+199	2.5E-154	1.7E+214
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNH	1.8E-313	1.4E-308	9.9E+86	7.2E-311	7.1E-235	3.5E-313	2.5E-308	9.9E+86
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
Strategy	9	10	11	12	13	14	15	
SKC	3.0E+180	5.0E+175	2.2E+161	9.8E+199	4.7E+257	1.9E+227	8.0E+165	
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SKB	4.7E+164	2.1E+214	3.0E-85	6.0E-154	2.0E+267	1.7E+243	2.3E+243	
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNH	1.6E-309	6.7E-314	6.2E-309	1.8E+175	4.6E-314	9.9E-307	5.1E-116	
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKC	3.0E-85	1.7E+262	3.1E+169	1.9E+214	1.8E+185	5.8E+252	2.0E+161	
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKB	1.8E-152	6.0E+247	5.8E+180	5.3E-85	3.4E-309	6.0E+197	3.0E-311	
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNH	1.2E-310	2.4E-196	5.2E-313	3.7E-308	9.9E+86	1.6E-310	8.2E-158	
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	

Table 71: Learned policy for G CO a by turn for 15 turn UrbanSim game using Q(λ), Boltzmann.

Strategy	1	2	3	4	5	6	7	8
SKC	6.3E-316	4.7E-316	1.2E+166	2.4E+232	1.4E-303	2.0E-316	6.0E-154	1.1E-152
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SKB	1.8E-152	1.7E+214	2.5E-154	1.4E+219	1.9E+219	4.7E+252	7.7E+170	7.3E+223
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNH	5.3E+228	3.0E+222	1.2E-259	2.1E-312	1.1E-311	2.7E-308	2.0E-309	2.1E-301
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKC	4.6E+228	1.3E-152	2.0E+161	3.3E+257	2.5E+198	1.7E-152	1.2E+214	4.0E+252
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKB	4.5E+257	5.8E+180	5.8E+252	1.8E+190	1.3E+213	5.9E+199	2.5E-154	1.7E+214
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNH	1.8E-313	1.4E-308	9.9E+86	7.2E-311	7.1E-235	3.5E-313	2.5E-308	9.9E+86
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
Strategy	9	10	11	12	13	14	15	
SKC	3.0E+180	5.0E+175	2.2E+161	9.8E+199	4.7E+257	1.9E+227	8.0E+165	
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SKB	4.7E+164	2.1E+214	3.0E-85	6.0E-154	2.0E+267	1.7E+243	2.3E+243	
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNH	1.6E-309	6.7E-314	6.2E-309	1.8E+175	4.6E-314	9.9E-307	5.1E-116	
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKC	3.0E-85	1.7E+262	3.1E+169	1.9E+214	1.8E+185	5.8E+252	2.0E+161	
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKB	1.8E-152	6.0E+247	5.8E+180	5.3E-85	3.4E-309	6.0E+197	3.0E-311	
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNH	1.2E-310	2.4E-196	5.2E-313	3.7E-308	9.9E+86	1.6E-310	8.2E-158	
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	

Table 72: Learned policy for H CO a by turn for 15 turn UrbanSim game using Q(λ), Boltzmann.

Strategy	1	2	3	4	5	6	7	8
SKC	6.3E-316	6.4E-316	1.2E+166	2.4E+232	1.4E-303	2.0E-316	6.0E-154	1.1E-152
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SKB	1.8E-152	1.7E+214	2.5E-154	1.4E+219	1.9E+219	4.7E+252	7.7E+170	7.3E+223
SNC	0.0E+00	2.8E+02	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNH	5.3E+228	3.0E+222	1.2E-259	2.1E-312	1.1E-311	2.7E-308	2.0E-309	2.1E-301
SNB	3.6E+02	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKC	4.6E+228	1.3E-152	2.0E+161	3.3E+257	2.5E+198	1.7E-152	1.2E+214	4.0E+252
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKB	4.5E+257	5.8E+180	5.8E+252	1.8E+190	1.3E+213	5.9E+199	2.5E-154	1.7E+214
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNH	1.8E-313	1.4E-308	9.9E+86	7.2E-311	7.1E-235	3.5E-313	2.5E-308	9.9E+86
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	2.7E+02	0.0E+00	0.0E+00	2.9E+02
Strategy	9	10	11	12	13	14	15	
SKC	3.0E+180	5.0E+175	2.2E+161	9.8E+199	4.7E+257	1.9E+227	8.0E+165	
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SKB	4.7E+164	2.1E+214	3.0E-85	6.0E-154	2.0E+267	1.7E+243	2.3E+243	
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNH	1.6E-309	6.7E-314	6.2E-309	1.8E+175	4.6E-314	9.9E-307	5.1E-116	
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKC	3.0E-85	1.7E+262	3.1E+169	1.9E+214	1.8E+185	5.8E+252	2.0E+161	
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKB	1.8E-152	6.0E+247	5.8E+180	5.3E-85	3.4E-309	6.0E+197	3.0E-311	
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNH	1.2E-310	2.4E-196	5.2E-313	3.7E-308	9.9E+86	1.6E-310	8.2E-158	
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	

Table 73: Learned policy for H CO b by turn for 15 turn UrbanSim game using Q(λ), Boltzmann.

Strategy	1	2	3	4	5	6	7	8
SKC	6.3E-316	6.4E-316	1.2E+166	2.4E+232	1.4E-303	2.0E-316	6.0E-154	1.1E-152
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SKB	1.8E-152	1.7E+214	2.5E-154	1.4E+219	1.9E+219	4.7E+252	7.7E+170	7.3E+223
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNH	5.3E+228	3.0E+222	1.2E-259	2.1E-312	1.1E-311	2.7E-308	2.0E-309	2.1E-301
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKC	4.6E+228	1.3E-152	2.0E+161	3.3E+257	2.5E+198	1.7E-152	1.2E+214	4.0E+252
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKB	4.5E+257	5.8E+180	5.8E+252	1.8E+190	1.3E+213	5.9E+199	2.5E-154	1.7E+214
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNH	1.8E-313	1.4E-308	9.9E+86	7.2E-311	7.1E-235	3.5E-313	2.5E-308	9.9E+86
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
Strategy	9	10	11	12	13	14	15	
SKC	3.0E+180	5.0E+175	2.2E+161	9.8E+199	4.7E+257	1.9E+227	8.0E+165	
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SKB	4.7E+164	2.1E+214	3.0E-85	6.0E-154	2.0E+267	1.7E+243	2.3E+243	
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNH	1.6E-309	6.7E-314	6.2E-309	1.8E+175	4.6E-314	9.9E-307	5.1E-116	
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKC	3.0E-85	1.7E+262	3.1E+169	1.9E+214	1.8E+185	5.8E+252	2.0E+161	
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKB	1.8E-152	6.0E+247	5.8E+180	5.3E-85	3.4E-309	6.0E+197	3.0E-311	
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNH	1.2E-310	2.4E-196	5.2E-313	3.7E-308	9.9E+86	1.6E-310	8.2E-158	
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	

Table 74: Learned policy for QRF by turn for 15 turn UrbanSim game using Q(λ), Boltzmann.

Strategy	1	2	3	4	5	6	7	8
SKC	6.3E-316	6.4E-316	1.2E+166	2.4E+232	1.4E-303	2.0E-316	6.0E-154	1.1E-152
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SKB	1.8E-152	1.7E+214	2.5E-154	1.4E+219	1.9E+219	4.7E+252	7.7E+170	7.3E+223
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
SNH	5.3E+228	3.0E+222	1.2E-259	2.1E-312	1.1E-311	2.7E-308	2.0E-309	2.1E-301
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKC	4.6E+228	1.3E-152	2.0E+161	3.3E+257	2.5E+198	1.7E-152	1.2E+214	4.0E+252
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MKB	4.5E+257	5.8E+180	5.8E+252	1.8E+190	1.3E+213	5.9E+199	2.5E-154	1.7E+214
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
MNH	1.8E-313	1.4E-308	9.9E+86	7.2E-311	7.1E-235	3.5E-313	2.5E-308	9.9E+86
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00
Strategy	9	10	11	12	13	14	15	
SKC	3.0E+180	5.0E+175	2.2E+161	9.8E+199	4.7E+257	1.9E+227	8.0E+165	
SKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SKB	4.7E+164	2.1E+214	3.0E-85	6.0E-154	2.0E+267	1.7E+243	2.3E+243	
SNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
SNH	1.6E-309	6.7E-314	6.2E-309	1.8E+175	4.6E-314	9.9E-307	5.1E-116	
SNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKC	3.0E-85	1.7E+262	3.1E+169	1.9E+214	1.8E+185	5.8E+252	2.0E+161	
MKH	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MKB	1.8E-152	6.0E+247	5.8E+180	5.3E-85	3.4E-309	6.0E+197	3.0E-311	
MNC	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	
MNH	1.2E-310	2.4E-196	5.2E-313	3.7E-308	9.9E+86	1.6E-310	8.2E-158	
MNB	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	0.0E+00	

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX H: STRATEGY LEVEL POLICIES BY ALGORITHM

URBANSIM

Table 75: Learned policy for all agents by turn for 15 turn UrbanSim game using Q(λ), ϵ-greedy.

STRATEGY	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
BATTALION COMMANDER	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB
CA UNIT	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB
E CO A	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB
E CO B	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB
F CO A	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB
F CO B	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB
G CO A	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB
G CO B	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB
H CO A	SNH	SNH	SNB	SNB	SNH	MNB	MKB	SKH	MNB	MKB	SKC	SNB	SKH	SKB	SNC
H CO B	SNH	SNH	SNB	SNB	SNH	MNB	MKB	SKH	MNB	MKB	SKC	SNB	SKH	SKB	SNC
QRF	SNH	SNH	SNB	SNB	SNH	MNB	MKB	SKH	MNB	MKB	SKC	SNB	SKH	SKB	SNC

Table 76: Learned policy for all agents by turn for 15 turn UrbanSim game using DQ-C, ϵ-greedy.

STRATEGY	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
BATTALION COMMANDER	MKH	SNB	MKH	MNC	SKC	MNH	MNH	SNH	SKB	MNC	MKH	SNB	SNH	MNH	SNC
CA UNIT	SNC	MKH	MKH	MNH	MKB	MKB	SNC	SNH	MNC	SKH	SNH	MNH	SKC	MKH	SNH
E CO A	SKH	SKC	SKH	MKC	MNB	SKB	MNC	MNH	MNB	MKC	MNC	MKH	MNH	MKB	SKH
E CO B	SNB	MKB	SNC	MKC	SNH	SNB	MKB	SKB	SNC	SNC	SNB	SKH	MNH	MKB	MNB
F CO A	SNB	SKC	SNH	SKC	SKB	SKB	SNC	SNB	MNH	MNH	MNC	SNC	MNC	SKH	MNC
F CO B	SNH	SKH	SKC	MKB	SNC	SKC	MKB	SNC	SNC	MNC	MKB	SNC	SNH	MKC	MKB
G CO A	SNB	SKB	MNC	MKB	SNH	SKB	MNH	SNH	MKC	SNH	SKB	SKC	SKC	MNC	SNB
G CO B	SNB	MNC	SNB	MNH	SNH	MKC	SKC	MNH	SKH	MKC	SNB	SKC	SKC	SNH	SNH
H CO A	SKC	MNH	MNH	SNB	SKH	SKB	SKH	SNB	SKB	MKH	SKC	SNB	SKC	SKH	SNC
H CO B	SNH	MNB	SNB	MKB	SKH	MKB	SKC	MNC	SNB	SKC	SNB	MNB	SKH	SKH	SNH
QRF	SNH	SKH	MNH	MKC	MNB	MKH	MNB	SKH	SKC	SNH	SKH	MNH	MKH	MNH	SKB

Table 77: Learned policy for all agents by turn for 15 turn UrbanSim game using DQ-C, Boltzmann.

STRATEGY	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
BATTALION COMMANDER	MKH	MNH	MNH	SKC	MKH	MKB	MKB	MNB	SNH	SKB	MNB	SKC	MKB	SKC	SNC
CA UNIT	MKB	SNC	MNB	SKC	MKB	MKH	SNB	MNH	SNH	SKC	MNH	SNC	SKC	SNH	MNH
E CO A	SNH	SKH	MKH	SNC	SKB	MKC	MNC	SKB	MKC	MNH	MKB	MNH	SNC	SNC	MKB
E CO B	MNB	SNH	MKC	SNC	MKH	SKC	SNB	MNH	SNH	SNB	SKC	MNH	SKB	MNC	SNC
F CO A	MNH	MNC	SKC	MKC	SNB	SKB	SKC	MKH	SNB	SNB	SNB	SNH	SNC	MKC	MNB
F CO B	MNH	MKB	SNB	SNC	MKH	SNH	MKH	MKH	SKH	MKH	SNH	SNC	MKB	MKC	MKH
G CO A	SNH	MKB	SNC	MNH	SNH	MNC	SKH	MNB	SKC	MNB	MKC	MNC	SNH	SKB	SNC
G CO B	MNH	SNC	MKB	SNC	MKB	MKB	SKB	SNH	MKC	SNH	SNH	MKC	MKB	SKB	MKB
H CO A	MKH	SNC	SKC	MKC	MNB	MKC	SKC	MNB	MNH	MNH	SKB	SKB	MNH	SKB	MKB
H CO B	SNB	SKC	MKB	SKB	SKH	SKC	MNB	MNC	MNC	SKB	SNC	SKB	SNB	MNC	SKH
QRF	SNB	SKB	SNB	SKB	SNB	MNC	SNB	MNC	SKH	SKB	MNB	SKH	MKH	SNC	SKB

Table 78: Learned policy for all agents by turn for 15 turn UrbanSim game using SARSA(λ), ϵ-greedy.

STRATEGY	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
BATTALION COMMANDER	MNC	MNB	SKC	SNH	SKC	SNH	SNB	SNB	SKC	MKC	MNC	MNC	SNC	SNC	SNC
CA UNIT	MNC	MNB	SKC	SNH	MNC	SNH	SNB	SNC	SKC	SKH	MNC	MNC	SNC	SNC	SNC
E CO A	MNC	MNB	SKC	SNH	MNC	SNH	SNB	SNC	SKC	SKH	MNC	MNC	SNC	SNC	SNC
E CO B	SNH	MNC	SNC	MKH	MNH	SKC	SNH	SKB	MNC	SKH	MKH	MNC	SKH	SKC	SKB
F CO A	SNH	MNC	SNC	MKH	MNH	SKC	SNH	SKB	MNC	SKH	MKH	MNC	SKH	SKC	SKB
F CO B	SNH	MNC	SNC	MKH	MNH	SKC	SNH	SKB	MNC	SKH	MKH	MNC	SKH	SKC	SKB
G CO A	SNH	MNC	SNC	MKH	MNH	SKC	SNH	SKB	MNC	SKH	MKH	MNC	SKH	SKC	SKB
G CO B	SNH	MNC	SNC	MKH	MNH	SKC	SNH	SKB	MNC	SKH	MKH	MNC	SKH	SKC	SKB
H CO A	SNH	MNC	SNC	MKH	MNH	SKC	SNH	SKB	MNC	SKH	MKH	MNC	SKH	SKC	SKB
H CO B	SNH	MNC	SNC	MKH	MNH	SKC	SNH	SKB	MNC	SKH	MKH	MNC	SKH	SKC	SKB
QRF	SNH	MNC	SNC	MKH	MNH	SKC	SNH	SKB	MNC	SKH	MKH	MNC	SKH	SKC	SKB

Table 79: Learned policy for all agents by turn for 15 turn UrbanSim game using SARSA(λ), Boltzmann.

STRATEGY	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
BATTALION COMMANDER	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC
CA UNIT	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC
E CO A	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC
E CO B	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC
F CO A	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC
F CO B	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC
G CO A	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC
G CO B	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC
H CO A	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC
H CO B	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC
QRF	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC	SKC

Table 80: Learned policy for all agents by turn for 15 turn UrbanSim game using Q(λ), Boltzmann.

STRATEGY	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
BATTALION COMMANDER	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB
CA UNIT	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB
E CO A	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB
E CO B	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB
F CO A	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB
F CO B	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB
G CO A	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB
G CO B	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB	MNB
H CO A	SNB	SNC	SKC	MKC	MNB	MKC	SKC	MNB	MNH	MNH	SKB	SKB	MNH	SKB	MKB
H CO B	SNB	SNC	SKC	MKC	MNB	MKC	SKC	MNB	MNH	MNH	SKB	SKB	MNH	SKB	MKB
QRF	SNB	SNC	SKC	MKC	MNB	MKC	SKC	MNB	MNH	MNH	SKB	SKB	MNH	SKB	MKB

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX I: ADDITIONAL BENCHMARKING RESULTS

This section provides additional insights into the performance of DQ-C in benchmarking environments.

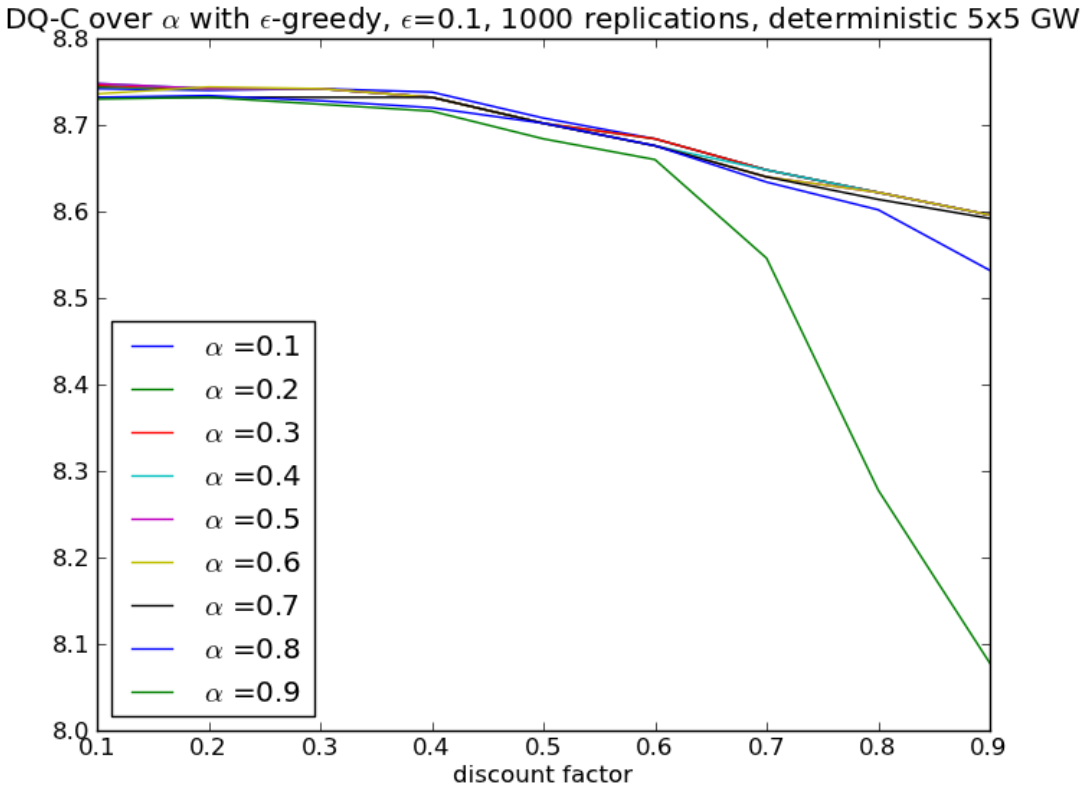


Figure 54: Performance of DQ-C in 5x5 deterministic grid world over γ by α paired with ϵ-greedy.

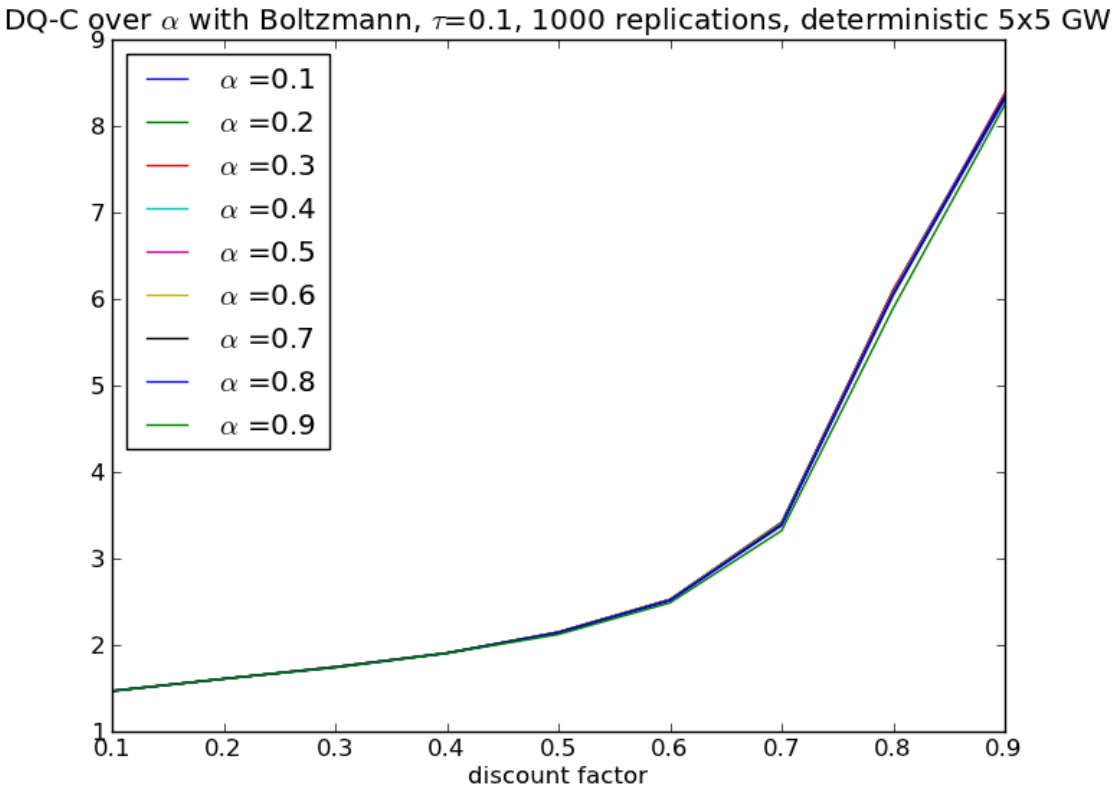


Figure 55: Performance of DQ-C in 5x5 deterministic grid world over γ by α paired with Boltzmann exploration.

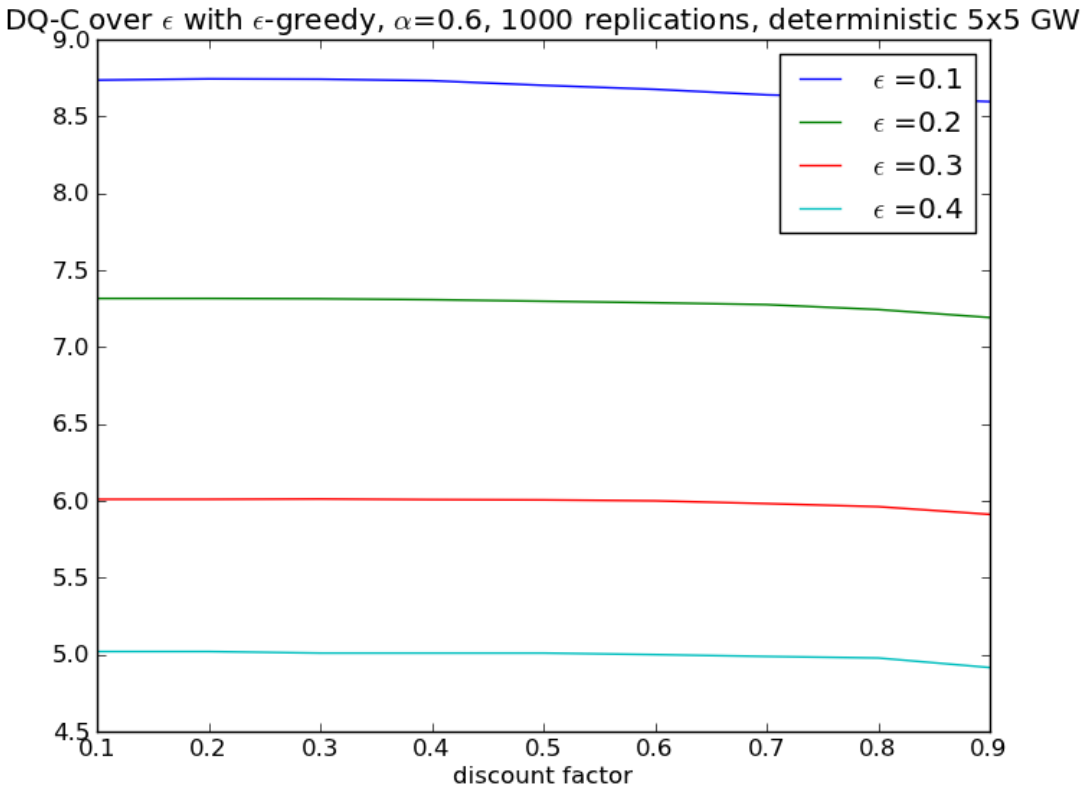


Figure 56: Performance of DQ-C in 5x5 deterministic grid world over γ by ϵ paired with ϵ-greedy with $\alpha = 0.6$.

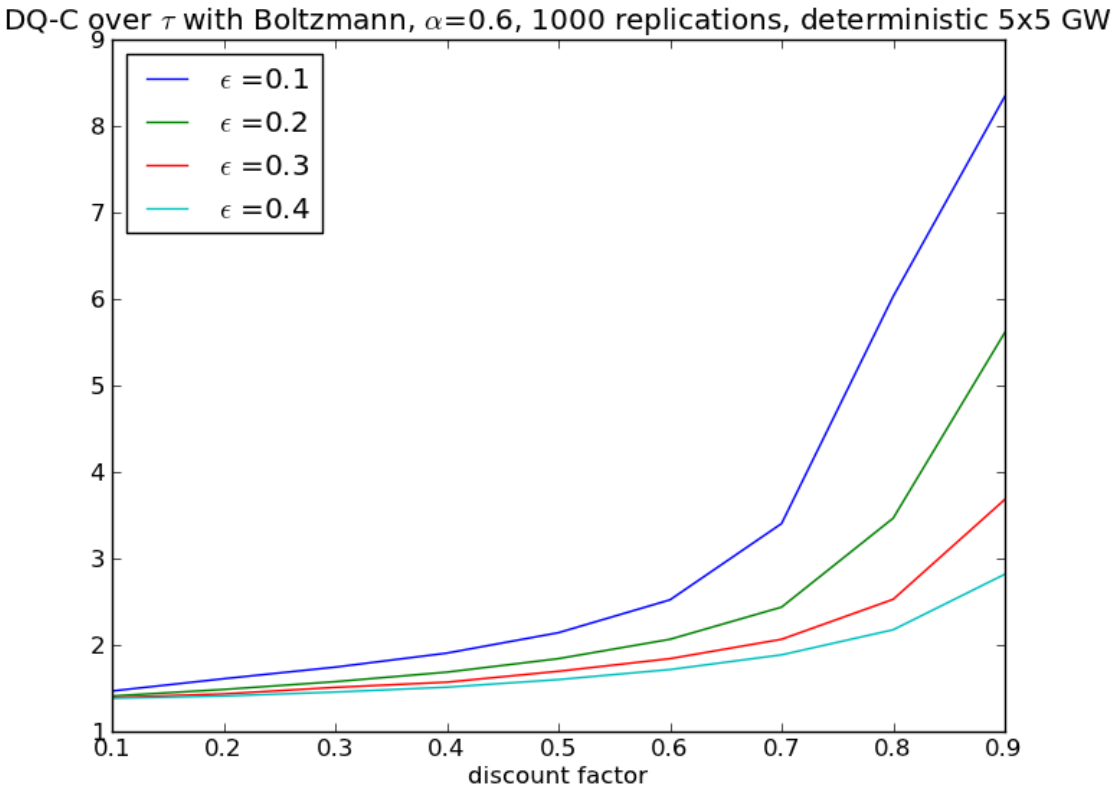


Figure 57: Performance of DQ-C in 5x5 deterministic grid world over γ by τ paired with Boltzmann exploration with $\alpha = 0.6$.

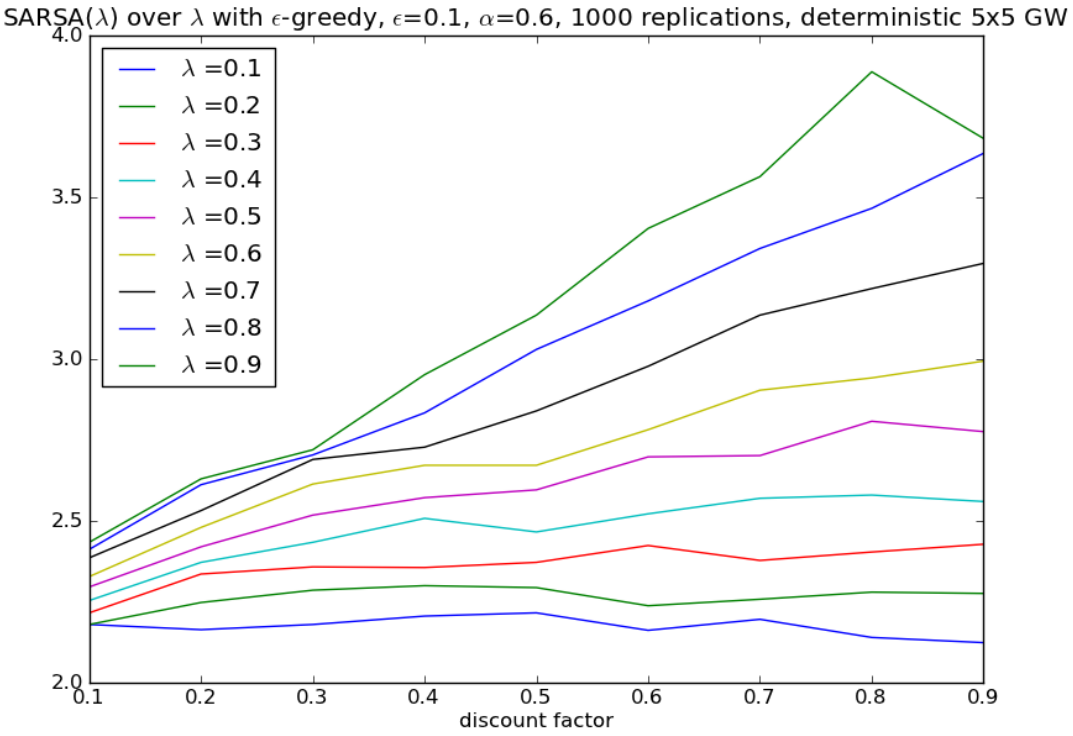


Figure 58: Performance of SARSA(λ) in 5x5 deterministic grid world over γ by λ paired with ϵ-greedy.

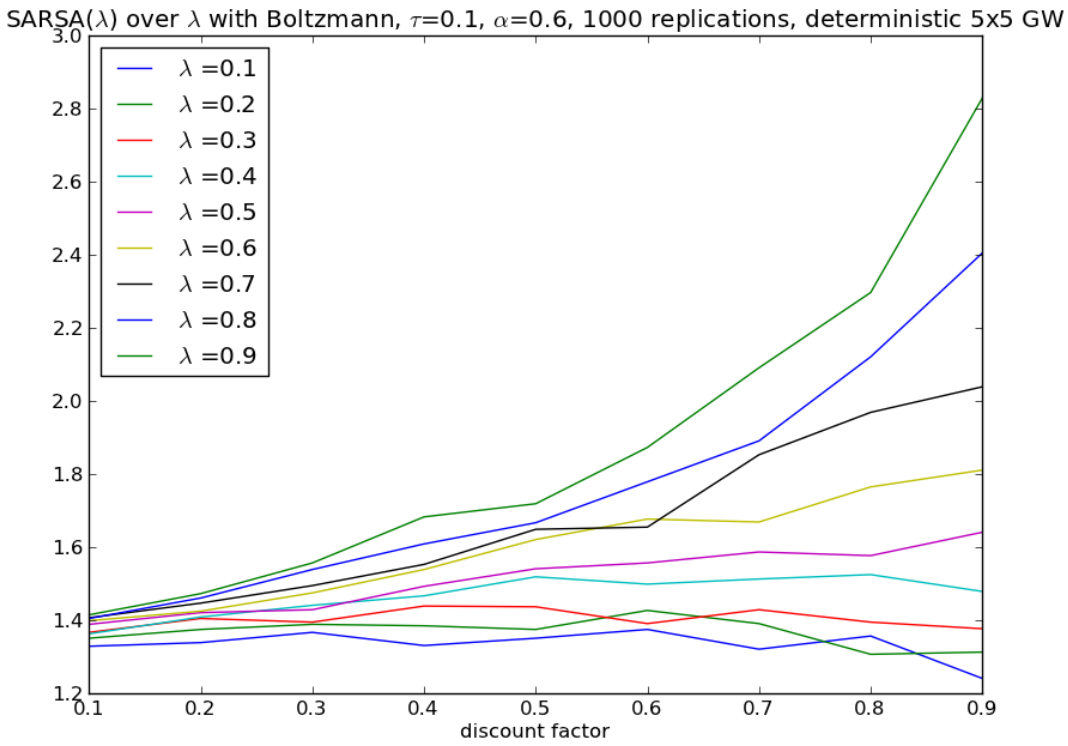


Figure 59: Performance of SARSA(λ) in 5x5 deterministic grid world over γ by λ paired with Boltzmann exploration.

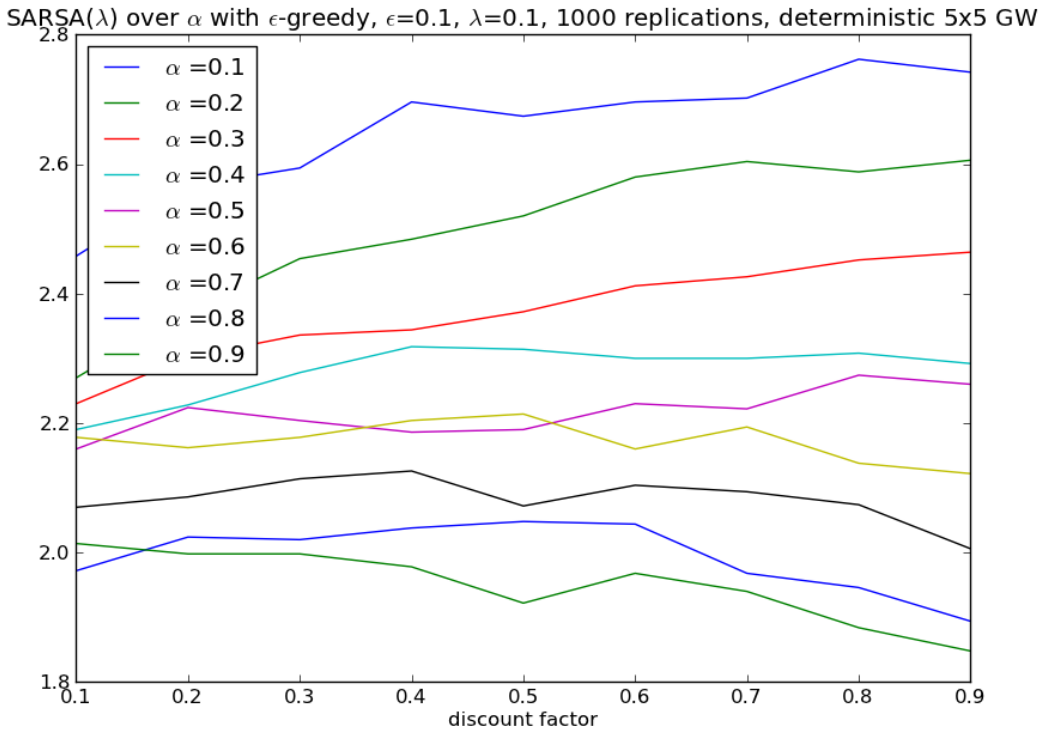


Figure 60: Performance of SARSA(λ) in 5x5 deterministic grid world over γ by α paired with ϵ-greedy with $\alpha = 0.6$.

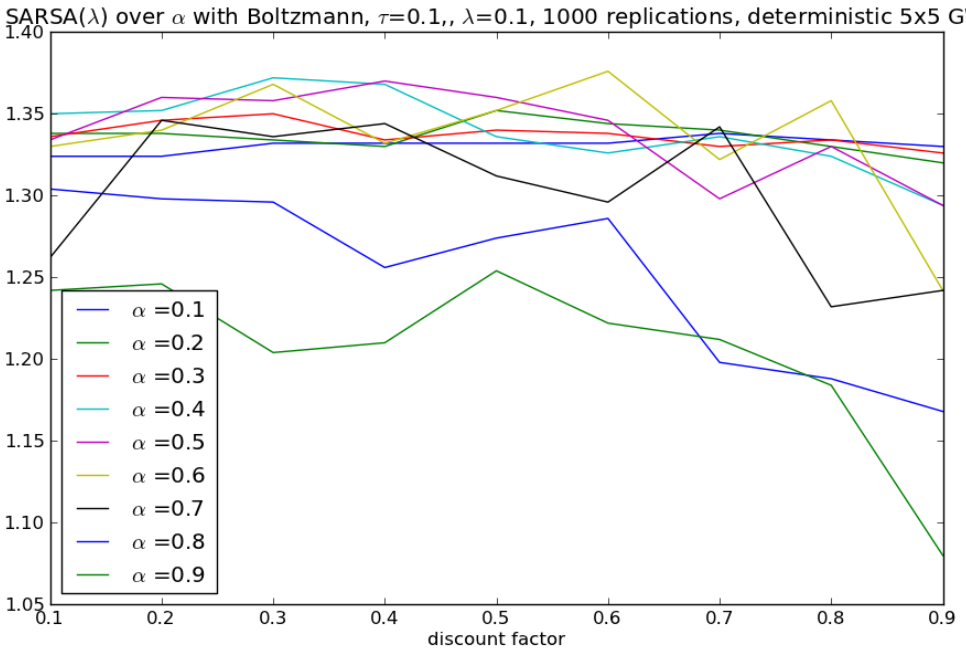


Figure 61: Performance of SARSA(λ) in 5x5 deterministic grid world over γ by τ paired with Boltzmann exploration with $\alpha = 0.6$.

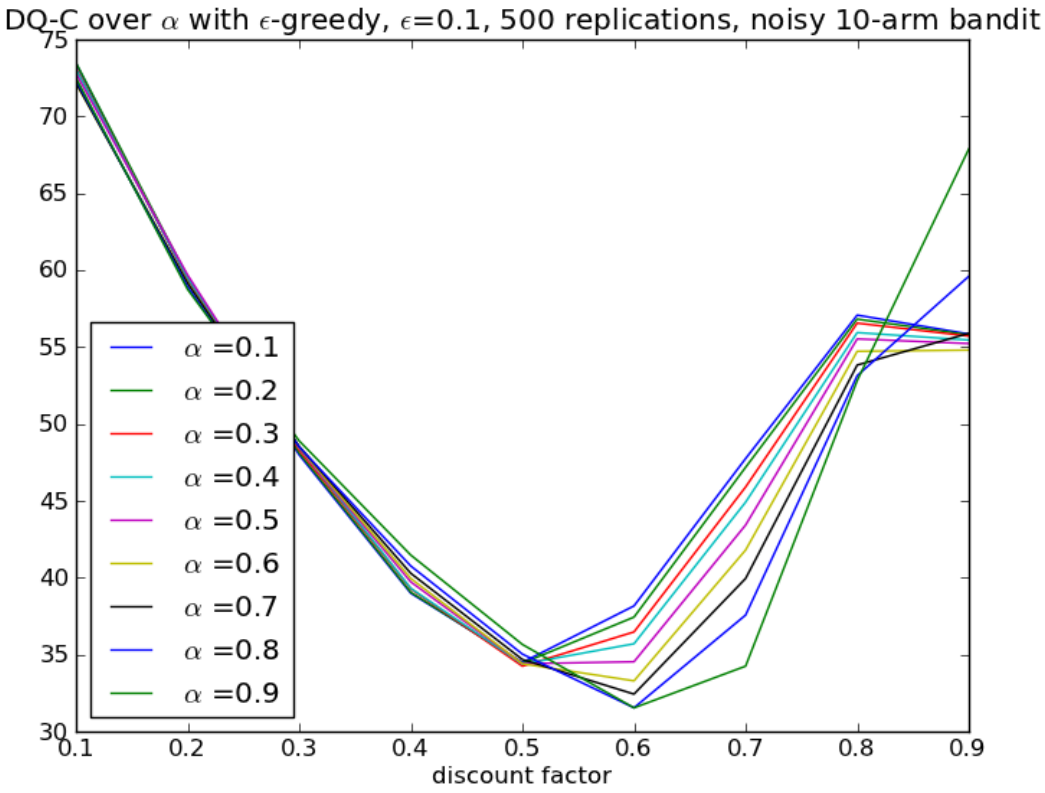


Figure 62: Performance of DQ-C in noisy 10-arm bandit over γ by α paired with ϵ-greedy.

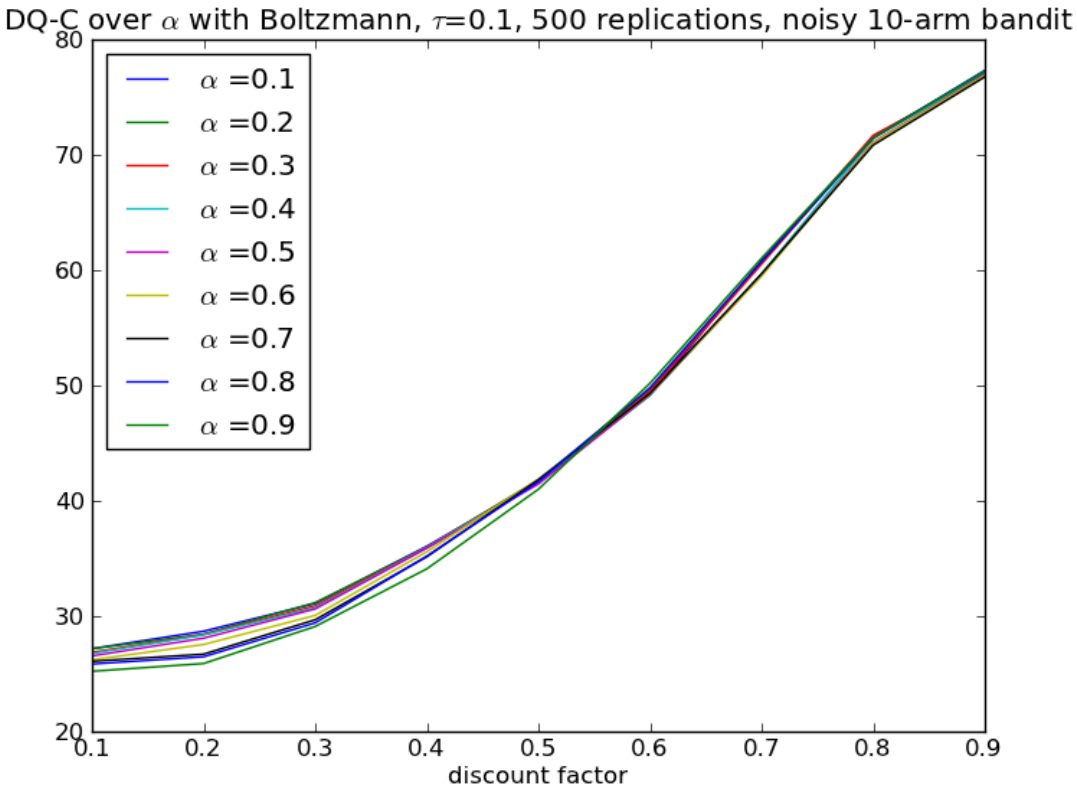


Figure 63: Performance of DQ-C in noisy 10-arm bandit over γ by α paired with Botzmann exploration.

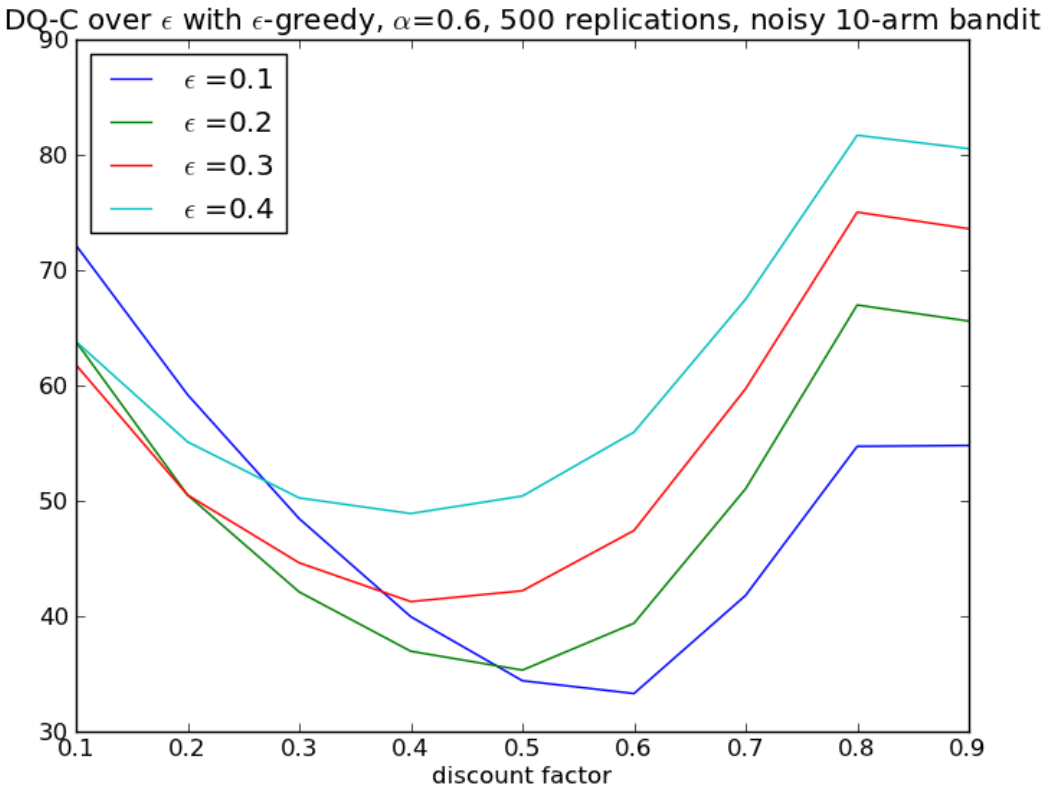


Figure 64: Performance of DQ-C in noisy 10-arm bandit over γ by τ paired with ϵ-greedy, $\alpha = 0.6$.

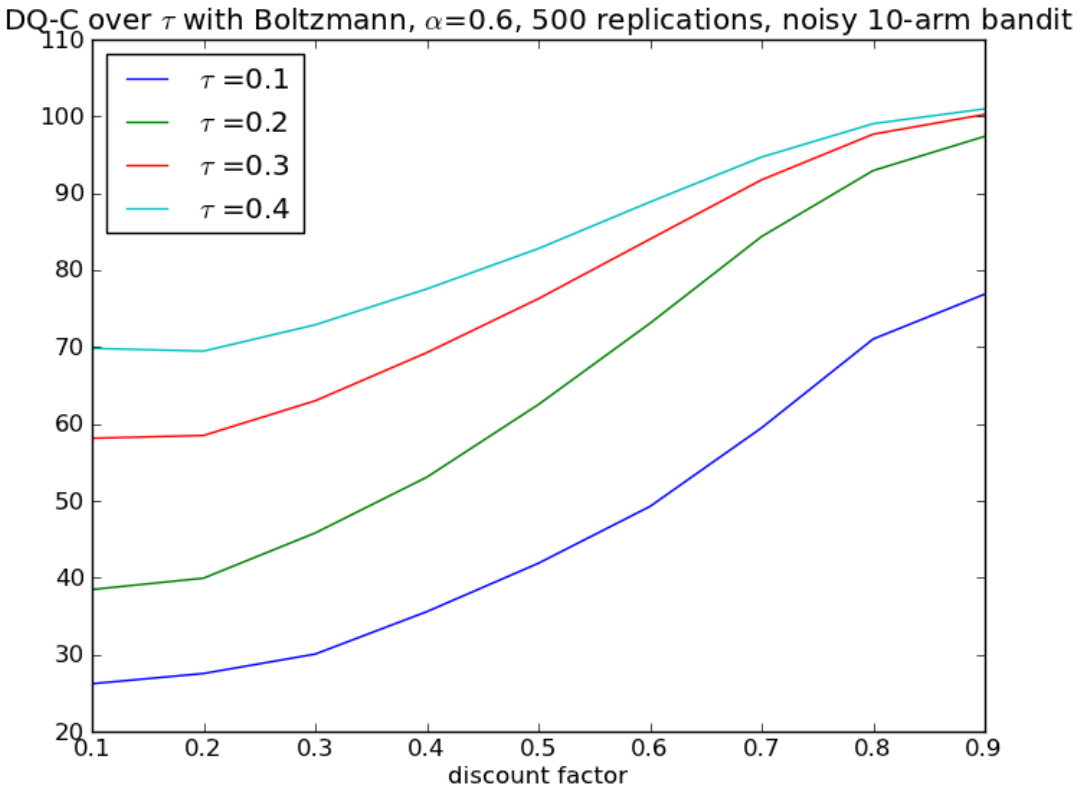


Figure 65: Performance of DQ-C in noisy 10-arm bandit over γ by τ paired with Botzmann exploration, $\alpha = 0.6$.

Table 81: Parameter settings that produced the best observed values for noisy 10-arm bandit.

ALGORITHM-π	α	γ	ϵ OR τ	λ
DQ-C, BOLTZMANN	0.9	0.1	0.1	NA
DQ-C, ϵ-GREEDY	0.9	0.6	0.1	NA
Q(λ), BOLTZMANN	0.2	0.8	0.1	0.4
Q(λ), ϵ-GREEDY	0.2	0.5	0.1	0.8
SARSA(λ), BOLTZMANN	0.9	0.5	0.3	0.8
SARSA(λ), ϵ-GREEDY	0.2	0.4	0.1	0.9

Table 82: Parameter settings that produced the best observed values for 5x5 Deterministic Gridworld.

ALGORITHM-π	α	γ	ϵ OR τ	λ
DQ-C, BOLTZMANN	0.1	0.5	0.1	NA
DQ-C, ϵ-GREEDY	0.1	0.4	0.1	NA
Q(λ), BOLTZMANN	0.9	0.5	0.4	0.3
Q(λ), ϵ-GREEDY	0.8	0.6	0.4	0.5
SARSA(λ), BOLTZMANN	0.6	0.9	0.1	0.9
SARSA(λ), ϵ-GREEDY	0.3	0.9	0.1	0.3

Table 83: Parameter settings that produced the best observed values for 5x5 Stochastic Gridworld.

ALGORITHM-π	α	γ	ϵ OR τ	λ
DQ-C, BOLTZMANN	0.7	0.9	0.1	NA
DQ-C, ϵ-GREEDY	0.8	0.4	0.1	NA
Q(λ), BOLTZMANN	0.9	0.9	0.2	0.5
Q(λ), ϵ-GREEDY	0.5	0.5	0.3	0.5
SARSA(λ), BOLTZMANN	0.6	0.9	0.1	0.9
SARSA(λ), ϵ-GREEDY	0.4	0.9	0.1	0.9

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Acuña, D., & Schrater, P. (2010). Structure learning in human sequential decision-making. *Computational Biology*, *6*(12), 1–12.

Ahner, D., Buss, A., & Ruck, J. (2006). Assignment scheduling capability for unmanned aerial vehicles. In L. Perrone, F. Wieland, J. Liu, B. Lawson, D. Nicol, & R. Fujimoto (Eds.), *Proceedings of the 39th conference on winter simulation* (pp. 1349–1356). Monterey, CA: IEEE Computer Society.

Ajzen, I. (1991). The theory of planned behavior. *Organizational Behavior and Human Decision Processes*, *50*(2), 179–211.

Alt, J., Jackson, L., Hudak, D., & Lieberman, S. (2009, December). The cultural geography model: Evaluating the impact of tactical operational outcomes on a civilian population in an irregular warfare environment. *Journal of Defense Modeling and Simulation*, *6*(4), 185–199.

Alt, J., Lieberman, S., & Blais, C. (2010). A use-case approach to the validation of social modeling and simulation. In *Springsim* (pp. 305–312). Orlando, FL: Society for Modeling and Simulation International.

Alt, J. K., Baez, F., & Darken, C. J. (2011, February). A practical situation based agent architecture for social simulations. In G. Jakobson, M. Endsley, & M. Kokar (Eds.), *Cognitive methods in situation awareness and decision support (CogSIMA), 2011 IEEE first international Multi-Disciplinary conference on* (pp. 305–312). Miami, FL: Institute of Electrical and Electronics Engineers (IEEE).

Anderson, J. (2005). *Cognitive psychology and its implications* (6th ed.). New York: Worth Publishers.

Anderson, J., & Schunn, C. (2005). Implications of the ACT-R learning Theory:No magic bullets. In *Advances in instructional psychology* (Vol. 5). Mahwah, NJ: Erlbaum.

Asmuth, J., & Littman, M. L. (2011). Learning is planning: near bayes-optimal reinforcement learning via monte-carlo tree search. In *Proceedings of the twenty-seventh conference annual conference on uncertainty in artificial intelligence* (pp. 19–26). Corvallis, Oregon: AUAI Press.

Audibert, J. Y., Munos, R., & Szepesvári, C. (2007). Tuning bandit algorithms in stochastic environments. In *Algorithmic learning theory* (pp. 150–165). Berlin: Springer.

Auer, P., & Ortner, R. (2010). UCB revisited: Improved regret bounds for the stochastic multi-armed bandit problem. *Periodica Mathematica Hungarica*, *61*(1), 55–65.

Bertsekas, D. (2011). *Dynamic programming and optimal control 3rd edition, volume II* (3rd ed.). Belmont, MA: Athena Scientific.

Bertsekas, D., & Tsitsiklis, J. (1996). *Neuro-dynamic programming* (No. 3). Belmont, MA: Athena Scientific.

Bertsekas, D. P. (1995). *Dynamic programming and optimal control* (2nd ed., Vol. 2). Belmont, MA: Athena Scientific.

Blum, A., & Monsour, Y. (2007). *Learning, regret minimization, and equilibria.* Available from http://repository.cmu.edu/compsci/133

Borrill, P. L., & Tesfatsion, L. S. (2010, July). *Agent-based modeling: The right mathematics for the social sciences?* (Tech. Rep.). Ames, Iowa: Iowa State University. Available from http://www2.econ.iastate.edu/tesfatsi/ABMRightMath.PBLTWP.pdf

Bouzy, B., & Chaslot, G. (2006). Monte-Carlo go reinforcement learning experiments. In *2006 IEEE symposium on computational intelligence and games* (pp. 187–194). Reno, Nevada.

Brenner, T. (2006). Agent learning representations. In L. Tesfastion (Ed.), *Handbook of computational economics: Agent-Based computational economics* (Vol. 2, pp. 896–942). Boston: Elsevier.

Brubaker, R., Loveman, M., & Stamatov, P. (2004). Ethnicity as cognition. *Theory and Society*, *33*(1), 31–64.

Buss, A. (2001). Basic event graph modeling. *Simulation News Europe*, *31*, 1–6.

Buss, A. (2002). Simkit: component based simulation modeling with simkit. In E. Yucesan (Ed.), *Proceedings of the 34th conference on winter simulation: Exploring new frontiers* (pp. 243–249). San Diego: IEEE Computer Society.

Buss, A. (2009). *Discrete event simulation modeling.* Unpublished course notes., Monterey, CA.

Buss, A., & Blais, C. (2007). Composability and component-based discrete event simulation. In S. Henderson (Ed.), *Proceedings of the 39th conference on winter simulation* (pp. 694–702). Washington, D.C.: IEEE Computer Society.

Buss, A. H., & Sanchez, P. J. (2002). Modeling very large scale systems: building complex models with LEGOs (Listener event graph objects). In E. Yucesan (Ed.), *Proceedings of the 34th conference on winter simulation: Exploring new frontiers* (pp. 732–737). San Diego: IEEE Computer Society.

Buss, A. H., & Sanchez, P. J. (2005). Simple movement and detection in discrete event simulation. In M. Kuhl (Ed.), *Proceedings of the 37th conference on winter simulation* (pp. 992–1000). Orlando, FL: IEEE Computer Society.

Cassenti, D. (2009, May). *Performance moderated function server's (pmfserv) military utility: A model and discussion* (Technical Report No. ARL-TR-4814). Adelphi, MD: Army Research Laboratory.

Cioppa, T. M., Lucas, T. W., & Sanchez, S. M. (2004). Military applications of agent-based simulations. In R. Ingalls (Ed.), *Proceedings of the 36th conference on winter simulation* (pp. 180–190). Washington, D.C.: IEEE Computer Society.

Cook, K., & Rice, E. (2006). Social exchange theory. In J. Delamater (Ed.), *Handbook of social psychology* (pp. 53–76). New York, NY: Springer.

Correll, S., & Ridgeway, C. (2006). Expectation states theory. In J. Delamater (Ed.), *Handbook of social psychology* (pp. 29–51). New York, NY: Springer.

Craig, C., Klein, M. I., Griswold, J., Gaitonde, K., McGill, T., & Halldorsson, A. (2012). Using cognitive task analysis to identify critical decisions in the laparoscopic environment. *Human Factors: The Journal of the Human Factors and Ergonomics Society*. Available from http://hfs.sagepub.com/content/early/2012/06/07/0018720812448393.abstract

DA. (1999). *Department of the army pamphlet (da pam) 5-11: Verification, validation, and accreditation of army models and simulations* (Tech. Rep.). Washington, D.C.: Department of the Army.

Davis, D. N., & Venkatamuni, V. M. (2010). A "Society of mind" cognitive architecture based on the principles of artificial economics. *International Journal of Artficial Life Research*, *1*(1), 51–71.

Dimitrakakis, C., & Lagoudakis, M. (2008). Rollout sampling approximate policy iteration. *Machine Learning*, *72*(3), 157–171.

DMSO. (2004). *The verification, validation, and accreditation recommended practices guide* (Tech. Rep.). Washington, D.C.: Department of Defense Modeling and Simulation Office.

Drogoul, A., & Ferber, J. (1994). *Multi-agent simulation as a tool for modeling societies: Application to social differentiation in ant colonies*. Berlin: Springer.

Duffy, J. (2006). Agent-based models and human subject experiments. In L. Tesfatsion & K. Judd (Eds.), *Handbook of computational economics: Agent-Based computational economics* (Vol. 2, chap. 19). Boston: Elsevier.

Epstein, J. (2006). *Generative social science*. Princeton, NJ: Princeton University Press.

Ferber, J., Gutknecht, O., & Michel, F. (2004). From agents to organizations: an organizational view of multi-agent systems. *Agent-Oriented Software Engineering IV*, 443–459.

Ferster, C., & Skinner, B. (1957). *Schedules of reinforcement*. East Norwalk, CT: Appleton-Century-Crofts.

Flavell, J. (1979). Metacognition and cognitive monitoring: A new area of cognitive-developmental inquiry. *American Psychologist*, *34*, 906–911.

Gilbert, N. (2004). Agent-based social simulation: dealing with complexity. *The Complex Systems Network of Excellence*, *9*(25), 1–14.

Gilbert, N. (2008). *Agent-based models* (No. 153). Los Angeles: Sage.

Goerger, S. R., McGinnis, M. L., & Darken, R. P. (2005, January). A validation methodology for human behavior representation models. *Journal of Defense Modeling and Simulation*, *2*(1), 5–17.

Hasselt, H. van. (2010). *Insights in reinforcement learning*. Unpublished doctoral dissertation, Dutch Research School for Information and Knowledge Systems, Netherlands.

Hogg, M. A. (2006). Intergroup relations. In J. Delamater (Ed.), *Handbook of social psychology* (pp. 479–501). New York, NY: Springer.

Howard, J., & Renfrow, D. (2006). Social cognition. In J. Delamater (Ed.), *Handbook of social psychology* (pp. 259–281). New York, NY: Springer.

Iiskala, T., Vauras, M., Lehtinen, E., & Salonen, P. (2011, June). Socially shared metacognition of dyads of pupils in collaborative mathematical problem-solving processes. *Learning and Instruction*, *21*(3), 379–393.

Ishida, F., Sasaki, T., Sakaguchi, Y., & Shimai, H. (2009, March). Reinforcement-learning agents with different temperature parameters explain the variety of human action-selection behavior in a markov decision process task. *Neurocomputing*, *72*(7-9), 1979–1984.

Jackson, L., & Sullivan, L. (1989). Cognition and affect in evaluations of stereotyped group members. *The Journal of Social Psychology*, *129*(5), 659–672.

John, B. E., & Kieras, D. E. (1996). The goms family of user interface analysis techniques: comparison and contrast. *ACM Transactions on Compututer and Human Interaction*, *3*(4), 320–351. Available from http://doi.acm.org/10.1145/235833.236054

Kaelbling, P., Littman, M., & Moore, A. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, *4*, 237–285.

Kakade, S. M., Lobel, I., & Nazerzadeh, H. (2010). *An optimal dynamic mechanism for Multi-Armed bandit processes* (Vol. abs/1001.4598; Tech. Rep.). Cornell University. Available from http://arxiv.org/abs/1001.4598

Kaplan, H. B. (2006). Social psychological perspectives on deviance. In J. Delamater (Ed.), *Handbook of social psychology* (pp. 451–478). New York, NY: Springer.

Kenrick, D. T., Griskevicius, V., Neuberg, S. L., & Schaller, M. (2010, May). Renovating the pyramid of needs: Contemporary extensions built upon ancient foundations. *Perspectives on Psychological Science*, *5*(3), 292–314. Available from http://pps.sagepub.com/lookup/doi/10.1177/1745691610369469

Klein, G. (1993). A recognition-primed decision (RPD) model of rapid decision making. In G. Klein, J. Orasanu, R. Calderwood, & C. Zsambok (Eds.), *Decision making in action: Models and methods* (pp. 138–147). Ablex Publishing.

Klein, G. (2008). Naturalistic decision making. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, *50*(3), 456-460. Available from http://hfs.sagepub.com/content/50/3/456.abstract

Kocsis, L., & Szepesvári, C. (2006). Bandit based monte-carlo planning. *Machine Learning: ECML 2006*, 282–293.

Kuleshov, V., & Precup, D. (2010). *Algorithms for the multi-armed bandit problem.* Available from http://www.cs.mcgill.ca/ vkules/

Kunde, D. (2005). *Event prediction for modeling mental simulation in naturalistic decision making*. Unpublished doctoral dissertation, Naval Postgraduate School.

Laird, J. E. (2008). Extending the soar cognitive architecture. In *Artificial general intelligence 2008: Proceedings of the first AGI conference* (pp. 224–235). Amsterdam, The Netherlands: IOS Press.

Laird, J. E., & Wray III, R. E. (2010). Cognitive architecture requirements for achieving agi. In *Proceedings of the 3rd conference on artificial general intelligence.* Lugano, Switzerland. Available from agi-conf.org/2010/wp-content/uploads/2009/06/paper$_4$.pdf

Langley, P., Laird, J. E., & Rogers, S. (2009, June). Cognitive architectures: Research issues and challenges. *Cognitive Systems Research*, *10*(2), 141–160. Available from http://linkinghub.elsevier.com/retrieve/pii/S1389041708000557

Lattal, K. A. (2010, January). DELAYED REINFORCEMENT OF OPERANT BEHAVIOR. *Journal of the Experimental Analysis of Behavior*, *93*(1), 129–139. Available from http://www.ncbi.nlm.nih.gov/pmc/PMC2801538

Lee, D., Seo, H., & Jung, M. W. (2012). Neural basis of reinforcement learning and decision making. *Annual Review of Neuroscience*, *35*(1), 287-308. Available from http://www.annualreviews.org/doi/abs/10.1146/annurev-neuro-062111-150512

Maio, G., Olson, J., Bernard, M., & Luke, M. (2006). Ideologies, values, attitudes, and behavior. In J. Delamater (Ed.), *Handbook of social psychology* (pp. 283–308). Springer.

Mansoor, P. (2007). A new counterinsurgency center of gravity analysis: Linking doctrine to action. *Professional Military Review, Journal of the U.S. Army*.

McKaughan, D. (2011). *Comparison of data development tools for populating cognitive models in social simulation*. Unpublished master's thesis, Naval Postgraduate School, Monterey, CA.

Miller, J., & Page, S. (2007). *Complex adaptive systems*. Princeton, NJ: Princeton University Press.

Moody, J., Liu, Y., Saffell, M., & Youn, K. (2004). Stochastic direct reinforcement: Application to simple games with recurrence. In *Proceedings of artificial multiagent learning. papers from the 2004 AAAI fall symposium, technical report FS-04* (Vol. 2). Arlington, VA: AAAI Press.

Moody, J., & Saffell, M. (2001). Learning to trade via direct reinforcement. *Neural Networks, IEEE Transactions on*, *12*(4), 875–889.

Nannini, C. (2006). *Analysis of the assignment scheduling capabiilty for unmanned aerial vehicles*. Unpublished master's thesis, Naval Postgraduate School, Monterey, CA.

Nargeot, R., & Simmers, J. (2011). Neural mechanism of operant conditioning and learning-induced behavioral plasticity in aplysia. *Cellular and Molecular Life Sciences*, *68*(5), 803–816.

Nedic, A., Tomlin, D., Holmes, P., Prentice, D. A., & Cohen, J. D. (2008, March). A decision task in a social context: Human experiments, models, and analyses of behavioral data. *Proceedings of the IEEE Conference on Decision and Control*, *100*(3), 713–733. Available from http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6069518

Nouris, A. (2010). *Efficient model-based exploration in continuous state-space environments*. Dissertation, Rutgers, The State University of New Jersey, Rutgers, New Jersey.

Okouchi, H. (2009, May). RESPONSE ACQUISITION BY HUMANS WITH DELAYED REINFORCEMENT. *Journal of the Experimental Analysis of Behavior*, *91*(3), 377–390. Available from http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2677568

Olson, J. R., & Olson, G. M. (1990, June). The growth of cognitive modeling in human-computer interaction since goms. *ACM Transactions on Compututer and Human Interaction*, *5*(2), 221–265. Available from http://dx.doi.org/10.1207/s15327051hci0502 3_4

Oyewole, S. A., & Haight, J. M. (2011). Determination of optimal paths to task goals using expert system based on goms model. *Computers in Human Behavior*, *27*(2), 823–833. Available from http://www.sciencedirect.com/science/article/pii/S074756321000350X

Ozcan, O., Alt, J., & Darken, C. (2011). Balancing exploration and exploitation in agent learning. In *Twenty-Fourth international FLAIRS conference.* Miami, FL: AAAI Press.

Ozkan, O. (2011). *Balancing exploration and exploitation in agent learning*. Unpublished master's thesis, Naval Postgraduate School, Monterey, CA.

Papadopoulos, M. (2010). *Reinforcement learning: A new approach for the cultural geography model*. Unpublished master's thesis, Naval Postgraduate School, Monterey, CA.

Papadopoulos, S., Alt, J., Darken, C., & Baez, F. (2013). Behavior selection using utility-based reinforcement learning in irregular warfare simulation models. *International Journal of Operations Research and Information Systems*, *4*(3), Manuscript accepted for publication.

Peeters, M., Könönen, V., Verbeeck, K., & Nowé, A. (2008). A learning automata approach to multi-agent policy gradient learning. In I. Lovrek (Ed.), *Knowledge-Based intelligent information and engineering systems* (pp. 379–390). Berlin: Springer.

Pollock, S., Alt, J., & Darken, C. (2011). Representing trust in cognitive social simulations. In J. Salerno (Ed.), *Social computing, behavioral-cultural modeling and prediction* (Vol. 6589, pp. 301–308). Berlin: Springer.

Powell, W. (2011). *Approximate dynamic programming: Solving the curses of dimensionality* (2nd ed.). Hoboken, NJ: John Wiley & Sons.

Powers, W., & Treval, W. (1973). *Behavior: The control of perception*. Chicago: Aldine.

Robbins, H. (1952). Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, *58*(5), 527–535.

Ross, S. (1982). *Introduction to stochastic dynamic programming*. Berkley, CA: Academic Press, Inc.

Russell, S., & Norvig, P. (2010). *Artificial intelligence: A modern approach* (Third ed.; M. Hirsch, Ed.). New York, NY: Prentice Hall.

Sato, M., & Kobayashi, S. (2000). Variance-penalized reinforcement learning for risk-averse asset allocation. In *Proceedings of the second international conference on intelligent data engineering and automated learning, data mining, financial engineering, and intelligent agents* (pp. 244–249). London, UK: Springer.

Schelling, T. C. (1972). Dynamic models of segregation. *Journal of Mathematical Sociology*, *1*, 143–186.

Shoham, Y., & Leyton-Brown, K. (2009). *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge: Cambridge University Press.

Skinner, B. (1938). *The behavior of organisms*. New York: Appleton-Century-Crofts.

Smith, A. (1966). *The wealth of nations*. Raleigh, NC: Hayes Barton Press.

Sorg, J., Singh, S., & Lewis, R. L. (2010). Variance-Based rewards for approximate bayesian reinforcement learning. In *Proceedings of the 26th conference on uncertainty in artificial intelligence* (pp. 564–571). Catalina Island, CA: AUAI Press.

Stets, J. (2006). Emotions and sentiments. In J. Delamater (Ed.), *Handbook of social psychology* (pp. 309–335). Berlin: Springer.

Steyvers, M., Lee, M., & Wagenmakers, E. (2009). A bayesian analysis of human decision-making on bandit problems. *Journal of Mathematical Psychology*, *53*(3), 168–179.

Still, S., & Precup, D. (2012, September). An information-theoretic approach to curiosity-driven reinforcement learning. *Theory in Bioscience*, *131*(3), 139–148.

Strehl, A., & Littman, M. (2004). An empirical evaluation of interval estimation for markov decision processes. In *Tools with artificial intelligence, 2004. ICTAI 2004. 16th IEEE international conference on tools with artificial intelligence* (pp. 128–135). Boca Raton, FL: IEEE Computer Society.

Strehl, A., & Littman, M. (2005). *A theoretical analysis of model-based interval estimation: Proofs* (Tech. Rep.). Rutgers, NJ: Rutgers University.

Sun, R. (2006). *Cognition and Multi-Agent interaction: From cognitive modeling to social simulation* (R. Sun, Ed.). Cambridge, MA: Cambridge University Press.

Sun, R. (2007a, September). Cognitive social simulation incorporating cognitive architectures. *IEEE Intelligent Systems*, *22*(5), 33–39.

Sun, R. (2007b). The importance of cognitive architectures: An analysis based on CLARION. *Journal of Experimental and Theoretical Artificial Intelligence*, *19*(2), 159–193.

Sun, R., & Naveh, I. (2007). Social institution, cognition, and survival: a cognitive–social simulation. *Mind & Society*, *6*(2), 115–142.

Sun, R., Zhang, X., & Mathews, R. (2006). Modeling meta-cognition in a cognitive architecture. *Cognitive Systems Research*, *7*(4), 327–338.

Sundaram, R. K. (2005). Generalized bandit problems. In D. Austen-Smith & J. Duggan (Eds.), *Social choice and strategic decisions* (pp. 131–162). Berlin: Springer.

Sutton, R., & Barto, A. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.

Sykulski, A. M., Adams, N. M., & Jennings, N. R. (2010). On-Line adaptation of exploration in the One-Armed bandit with covariates problem. In S. Draghici, T. M. Khoshgoftaar, V. Palade, W. Pedrycz, M. A. Wani, & X. Zhu (Eds.), *Proceedings of the internal conference of machine learning and applications* (pp. 459–464). Washington, D.C.: IEEE Computer Society.

Szepesvari, C. (2010). *Algorithms for reinforcement learning*. San Francisco: Morgan & Claypool Publishers.

Szita, I., Chaslot, G., & Spronck, P. (2010). Monte-carlo tree search in settlers of catan. *Advances in Computer Games*, 21–32.

Taatgen, N., Lebiere, C., & Anderson, J. (2006). Modeling paradigms in ACT-R. In R. Sun (Ed.), *Cognition and multi-agent interaction: From cognitive modeling to social simulation* (pp. 29–52). Cambridge: Cambridge University Press.

Taatgen, N. A., & Anderson, J. R. (2008). Constraints in cognitive architectures. In *Cambridge handbook of computational psychology* (pp. 170–185). Cambridge: Cambridge University Press.

Tesfatsion, L., & Judd, K. (Eds.). (2006). *Handbook of computational economics: Agent-Based computational economics* (Vol. 2) (No. 13). Boston: Elsevier.

Thorndike, E. (1911). *Animal intelligence*. New York: MacMillan.

Thrun, S., & Schwartz, A. (1993). Issues in using function approximation for reinforcement learning. In *Proceedings of the 1993 connectionist models summer school.* Hillsdale, NJ: Erlbaum.

Tokic, M. (2010). Adaptive epsilon-greedy exploration in reinforcement learning based on value differences. In *KI 2010: Advances in artificial intelligence* (pp. 203–210). Berlin: Springer.

Tran-Thanh, L., Chapman, A., Munoz De Cote Flores Luna, J., Rogers, A., & Jennings, N. (2010). Epsilon–First policies for Budget–Limited Multi-Armed bandits. In *Aaai conference on artificial intelligence.* Palo Alto, CA: AAAI Press.

Valgaeren, K., Croonenborghs, T., & Colleman, P. (2009). *Reinforcement learning with monte carlo tree search.* Available from http://users.telenet.be/KimValgaeren/eindwerk20ppt/Paper20Valgaeren20Kim.pdf

Walsh, M., & Anderson, J. (2010). Neural correlates of temporal credit assignment. In D. Salvucci (Ed.), *Proceedings of the 10th international conference on cognitive modeling* (pp. 265–270). Philadelphia, PA: Drexell University.

Wang, N., Pynadath, D., & Marsella, S. (2012). Toward automatic verification of multiagent systems for training simulations. In *Intelligent tutoring systems* (pp. 151–161). Berlin: Springer. Available from http://www.springerlink.com/index/E46382Q47W5H1676.pdf

Wansbury, T., Hart, J., Gordon, A. S., & Wilkinson, J. (2010). UrbanSim: training adaptable leaders in the art of battle command. In *The Interservice/Industry training, simulation and education conference (I/ITSEC)* (Vol. 2010). Orlando, FL. Available from http://ntsa.metapress.com/index/V0018279L8J824N8.pdf

Watkins, C., & Dayan, P. (1992). Q-learning. *Machine Learning*, *8*(3), 279–292.

Wickens, C., & Hollands, J. (2000). *Engineering psychology and human performance* (3rd ed.). Upper Saddle River, New Jersey: Prentice Hall.

Wilson, R. A. (2001). Group-level cognition. *Philosophy of Science*, *68*(3), 262–273.

Wray, R., & Jones, R. (2006). Considering SOAR as an agent architecture. In R. Sun (Ed.), *Cognition and Multi-Agent interaction: From cognitive modeling to social simulation* (pp. 53–78). Cambridge: Cambridge University Press.

Yi, M. S., Steyvers, M., & Lee, M. (2009). Modeling human performance in restless bandits with particle filters. *The Journal of Problem Solving*, *2*(2), 5.

Yu, H., & Bertsekas, D. (2011). *Q-Learning and policy iteration algorithms for stochastic shortest path problems* (Tech. Rep.). Massachusetts Institute of Technology.

Zacharias, G., MacMillan, J., & Van Hemel, S. E. (2008). *Behavioral modeling and simulation: From individuals to societies* (G. Zacharias, J. MacMillan, & S. Van Hemel, Eds.). Washington, D.C.: National Academies Press.

INITIAL DISTRIBUTION LIST

1.Defense Technical Information Center
Ft. Belvoir, Virginia

2.Dudley Knox Library
Naval Postgraduate School
Monterey, California

3.Dr. Christian J. Darken
Naval Postgraduate School
Monterey, California

4.Dr. Jeffrey Appleget
Naval Postgraduate School
Monterey, California

5.Dr. Arnold Buss
Naval Postgraduate School
Monterey, California

6.Dr. Michael McCauley
Naval Postgraduate School
Monterey, California

7.Dr. Michael Jaye
Naval Postgraduate School
Monterey, California