

AI-Based Autonomous Broker for Smart Grids: Theory, Design and Practice

Thesis submitted in partial
fulfillment of the requirements of the degree of

Doctor of Philosophy
in
Computer Science and Engineering
by Research

by

Sanjay Chandlekar

2020701017

sanjay.chandlekar@research.iiit.ac.in

Advised by Dr. Sujit P Gujar



Machine Learning Lab
International Institute of Information Technology, Hyderabad
(Deemed to be University)
Hyderabad - 500032, India
May, 2025

Copyright © Sanjay Chandlekar, 2025

All Rights Reserved

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "**AI-Based Autonomous Broker for Smart Grids: Theory, Design and Practice**" by **Sanjay Chandlekar**, has been carried out under my supervision and is not submitted elsewhere for a degree.

20/05/2025

Date

Adviser: Dr. Sujit P Gujar

To, my parents and sister,

*who were always there by my side and assured me of their everlasting love and support throughout
the challenging game called life.*

Acknowledgements

I would like to express my deepest gratitude to my advisor Prof. Sujit Gujar for his invaluable support and guidance. I will always cherish the kind of support and encouragement I received from him that made my journey delightful and memorable. His working style and expert guidance have positively impacted me and made me a better researcher. I will also remember our fun conversations on various topics over tea breaks. His continual support, wise suggestions, and belief in my ability have been instrumental in seeing this work through and writing my thesis. It truly has been an honor working with him. Many thanks to Prof. Praveen Paruchuri for being supportive during the PowerTAC work; his insightful observations and ideas have really helped to achieve great performance in the PowerTAC tournaments.

I am extremely grateful to Dr. Easwar Subramanian and Dr. Sanjay Bhat for their valuable ideas and advice. I am grateful for having known both of them before joining IIIT Hyderabad, the understanding between us made our PowerTAC collaboration fun to work with. I also want to particularly thank Dr. Easwar for his guidance and support since the time of TCS. It is always insightful to discuss things with him, whether professionally or personally. I also want to extend my sincere gratitude to Prof. Shweta Jain for her knowledgeable insights during the meetings as well as her support during the paper writing. I would like to thank all the professors at MLL for always being supportive. I am thankful to IIIT Hyderabad for providing me with this opportunity. I would like to thank the support staff for providing lab spaces, accommodation, food, and other necessary things during my stay on campus; their hard work has made my stay at IIIT Hyderabad comfortable. I

am grateful to all IIIT Hyderabad and Nirma University professors who have taught me and helped me improve my knowledge of the subject. I would also like to thank the IIIT administrators for their timely support with administrative procedures, TA stipends, RA stipends, conference travel documentation.

I am thankful to Bharat Electronics Limited for providing me the financial and logistic support. Many thanks to Microsoft Research, ACM India, AAMAS and IJCAI as well for providing grants for conference travel and attending prestigious conferences.

This endeavour would not have been possible without the constant support and encouragement from my friends, Sahishnavi, Sagar, Pawan, and Samriddhi. They made this journey memorable, and I will cherish the time I have spent with them. I would be remiss in not mentioning all my friends, Vaibhav, Srikar, Nikhil, Jigyasu, Ashish, Abhinaba, Prasha, Krutika, Chetan, Arpit, Prayushi, for making the IIIT life absolute fun. I am thankful to all my friends at MLL, Sankarshan, Manisha, Samhita, Shaily, Varul, Siddharth, Sambhav, and Shantanu, for making the lab environment so cheerful and friendly.

Words cannot express my gratitude to my parents, Meenaben and Rajendrabhai, and sister Nisha, without whom I could not have completed this journey. I am grateful to them for always believing in my abilities, providing unwavering support, and being there with me in tough times. I would like to thank all my friends for their love and encouragement throughout this journey. Finally, I would like to thank everyone who has made my stay at IIIT Hyderabad the most memorable. Thank you, everyone, for supporting me and being a part of my academic journey.

Abstract

The emergence of AI-driven systems has transformed smart grid networks, enabling widespread automation in decision-making. A smart grid is an advanced electricity distribution system that empowers customers to actively participate through smart meters. These grids operate across three key markets: wholesale, tariff, and balancing markets. In this ecosystem, distribution companies, or electricity brokers, play a pivotal role by procuring electricity in the wholesale market through double auctions and selling it to customers in the tariff market. Meanwhile, the balancing market oversees real-time supply and demand, penalizing brokers responsible for imbalances. The primary goal of brokers is to maximize profits, which involves addressing three critical challenges: (i) minimizing procurement costs in the wholesale market, (ii) offering competitive yet profitable tariffs in the tariff market, and (iii) mitigating peak demand scenarios to avoid penalties. Achieving this requires brokers to develop intelligent strategies leveraging AI techniques. This thesis develops AI-based strategies that enhance the efficiency of electricity brokers, tested using a close-to-real-world simulation platform, PowerTAC.

Specifically, we design various bidding strategies for brokers operating in the wholesale market, where electricity is traded through day-ahead periodic double auctions (PDAs). The first strategy draws inspiration from game theory, leveraging Nash equilibrium principles for a single-buyer and single-seller setup. This is extended to real-world scenarios consisting of multiple buyers and sellers using reinforcement learning techniques. The second strategy reduces procurement costs by exploiting the supply curve information of prominent sellers to inform bidding decisions. The third strategy employs Markov Perfect

Nash Equilibrium (MPNE) policies to model buyer behavior with known supply curves and introduces an algorithm to address scenarios where supply curve information is unavailable. The fourth strategy leverages Monte Carlo Tree Search (MCTS) to operate in the continuous action space of bid prices for optimized bidding in PDAs.

In the tariff market, we develop strategies for generating attractive tariff contracts to build and retain a robust customer base. Using game theory, we demonstrate that maintaining an optimal market share significantly boosts net revenue. To achieve this, we propose both heuristic and learning-based approaches, with the latter employing multi-armed bandit (MAB) techniques for tariff generation.

To address peak demand scenarios, we propose a demand response mechanism that incentivizes customers to shift their electricity usage away from peak hours. We present an optimal algorithm for allocating discounts that maximize expected peak demand reduction while adhering to budget constraints. Additionally, we introduce an MAB-based online algorithm to handle cases where customer responses to incentives are initially unknown.

Finally, we integrate these strategies to develop our autonomous broker, VidyutVanika, for the PowerTAC simulation-based tournament. VidyutVanika competed against other brokers with the objective of maximizing profits. The results from the PowerTAC 2021 and 2022 tournaments demonstrate the effectiveness of our approach, with VidyutVanika emerging as the champion in both years.

Research Papers from the Thesis Work

Journals

1. **Sanjay Chandlekar**, Bharat Manvi, Easwar Subramanian, and Sujit Gujar. “Equilibrium Analysis and Strategic Bidding for Buyers in Multi-unit Periodic Double Auctions.” In Artificial Intelligence Journal, **(AIJ) (Under Review)**

Book Chapters

1. **Sanjay Chandlekar**, Bala Suraj Pedasingu, Susobhan Ghosh, Easwar Subramanian, Sanjay Bhat, Praveen Paruchuri, and Sujit Gujar. “VidyutVanika: AI-Based Autonomous Broker for Smart Grids - From Theory to Practice.” In the book “Energy Sustainability through Retail Electricity Markets: The Power Trading Agent Competition (Power TAC) Experience.” Springer International Publishing, Nov. 2023

Conference Papers

1. **Sanjay Chandlekar.** “Towards Revolutionized Smart Grids: An AI-Driven Broker for Improved Operational Efficiency.” In Proceedings of the 33rd International Joint Conference on Artificial Intelligence, Doctoral Consortium Track, August 2024 (*IJCAI 2024 DC, CORE A**)
2. Bharat Manvi, **Sanjay Chandlekar** and Easwar Subramanian. “Optimizing Prosumer Policies in Periodic Double Auctions Inspired by Equilibrium Analysis.” In Proceedings of the 33rd International Joint Conference on Artificial Intelligence, August 2024 (*IJCAI 2024, CORE A**)
3. **Sanjay Chandlekar** and Easwar Subramanian. “A Novel Bidding Strategy for PDAs using MCTS in Continuous Action Spaces.” In Proceedings of the 12th International Workshop on Engineering Multi-Agent Systems, Lecture Notes in Computer Science(), vol 15152. Springer, May 2024 (*EMAS@AAMAS 2024*)
4. **Sanjay Chandlekar**, Shweta Jain, and Sujit Gujar. “A Novel Demand Response Model and Method for Peak Reduction in Smart Grids - PowerTAC.” In Proceedings of the 32st International Joint Conference on Artificial Intelligence, August 2023 (*IJCAI 2023, CORE A**)
5. **Sanjay Chandlekar**, Arthik Boroju, Shweta Jain, and Sujit Gujar. “A Novel Demand Response Model and Method for Peak Reduction in Smart Grids - PowerTAC.” In Proceedings of the 22nd International Conference on Autonomous Agents and Multiagent Systems, May 2023 (*AAMAS 2023, CORE A**)
6. **Sanjay Chandlekar**, Easwar Subramanian, and Sujit Gujar. “Multi-armed Bandit Based Tariff Generation Strategy for Multi-Agent Smart Grid Systems.” In Proceedings of the 11th International Workshop on Engineering Multi-Agent Systems. Lec-

ture Notes in Computer Science(), vol 14378. Springer, May 2023 (***EMAS@AAMAS 2023***)

7. **Sanjay Chandlekar**, Bala Suraj Pedasingu, Easwar Subramanian, Sanjay Bhat, Praveen Paruchuri, and Sujit Gujar. “VidyutVanika21: An Autonomous Intelligent Broker for Smart-grids.” In Proceedings of the 31st International Joint Conference on Artificial Intelligence, July 2022 (***IJCAI 2022, CORE A****)
8. **Sanjay Chandlekar**, Easwar Subramanian, Sanjay Bhat, Praveen Paruchuri, and Sujit Gujar. “Multi-unit Double Auctions: Equilibrium Analysis and Bidding Strategy using DDPG in Smart-grids.” In Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems, May 2022 (***AAMAS 2022, CORE A****)

Releases

1. **VidyutVanika21** broker binary (PowerTAC2021 Tournament Winning Codebase)
URL: <https://github.com/magnetar-iiith/POWERTAC/tree/VidyutVanika21>
2. **VidyutVanika22** broker binary (PowerTAC2022 Tournament Winning Codebase)
URL: <https://github.com/magnetar-iiith/POWERTAC/tree/VidyutVanika22>

Contents

Chapter	Page
1 Introduction	1
1.1 Motivation to Use Smart Grids	5
1.1.1 From Economical Point of View	5
1.1.2 From Customers' Point of View	6
1.2 Smart Grid Systems and Its Challenges	8
1.3 Problems Addressed	10
1.4 Proposed Approaches	11
1.4.1 Part A: In the Wholesale Market of a Smart Grid	11
1.4.1.1 Bidding Strategy via Equilibrium Theory	12
1.4.1.2 Bidding Strategy via Learning Supply Curve	12
1.4.1.3 Bidding Strategy via Markov Perfect Nash Equilibrium . .	13
1.4.1.4 Bidding Strategy using MCTS in Continuous Action Spaces	13
1.4.2 Part B: In the Tariff Market of a Smart Grid	13
1.4.3 Part B ₁ : Tariff Contract Generation	14
1.4.3.1 Tariff Contract Generation via Heuristic Method	14
1.4.3.2 Tariff Contract Generation via Contextual-MAB Method .	14
1.4.4 Part B ₂ : Peak Reduction via Demand Response	15
1.4.4.1 Demand Response Model and Method for Peak Reduction	15
1.5 Outline of the Work	15
2 Game Theory and Mechanism Design	18
2.1 Game Theory: An Overview	19
2.1.1 What is Game Theory? Definitions and Examples	20
2.1.2 Preliminaries and Properties of a Game	24
2.2 Mechanism Design: An Overview	33
2.2.1 Preliminaries	35
2.2.2 Important Definitions in Mechanism Design	36
2.2.3 Auctions	44
2.2.3.1 Auction Types and Desirable Properties	44

2.2.3.2	Single Sided Auctions	46
2.2.3.3	Double Sided Auctions	51
2.3	Summary	56
3	Reinforcement Learning	57
3.1	Reinforcement Learning: An Overview	58
3.2	Prelimanaries and Definitions	63
3.3	Tabular RL Methods	66
3.3.1	Dynamic Programming	67
3.3.1.1	Policy Iteration	67
3.3.1.2	Value Iteration	70
3.3.2	Monte Carlo	71
3.3.3	Temporal-Difference Learning	73
3.3.3.1	SARSA: On-Policy TD Control	74
3.3.3.2	Q-Learning: Off-Policy TD Control	76
3.4	Approximate Solution Methods	77
3.4.1	Policy Gradient Methods	78
3.4.1.1	Deep Deterministic Policy Gradient (DDPG)	79
3.5	Monte-Carlo Tree Search: An Overview	82
3.5.1	MCTS for Discrete Action Spaces	83
3.5.2	MCTS for Continuous Action Spaces: Simple Progressive Widening (SPW)	84
3.6	Multi-Armed Bandits and its Derivatives: An Overview	85
3.6.1	Contextual MAB	86
3.6.2	MAB Algorithms	89
3.6.2.1	The Epsilon-Greedy Algorithm	89
3.6.2.2	The UCB Algorithm	91
3.6.2.3	The EXP-3 Algorithm	93
3.7	Summary	96
4	Smart Grids: An Overview, PowerTAC Simulation, and Literature Review	97
4.1	Introduction to a Smart Grid System	98
4.2	The PowerTAC Simulator	101
4.2.1	Information from the Simulator	102
4.2.2	The PowerTAC Wholesale Market	105
4.2.2.1	k -Double Auction	105
4.2.3	The PowerTAC Tariff Market	107
4.2.4	The PowerTAC Balancing Market	109
4.2.5	The PowerTAC Broker	110
4.2.6	The PowerTAC Competition	110
4.3	A Typical Broker's Activities in the PowerTAC	112

4.3.1	A Broker in the Wholesale Market	113
4.3.2	A Broker in the Tariff Market	113
4.3.3	A Broker in the Balancing Market	114
4.4	Liturature Survey	114
4.4.1	Bidding Strategies in Double Auctions	114
4.4.2	In the PowerTAC Wholesale Market	117
4.4.3	In the PowerTAC Tariff Market	129
4.4.4	Demand Response for Peak Reduction	134
4.5	Summary	136
5	Designing Bidding Strategies via Equilibrium Theory	138
5.1	Introduction	139
5.2	Preliminaries	141
5.3	Equilibrium Analysis for Multi-unit Double Auction	143
5.3.1	Special case of $n = 1$ and $k = 0.5$	152
5.3.2	Special case of $n = 2$ and $k = 0.5$	153
5.3.3	Generalizing the Scale factors Calculation	160
5.4	Extending Theory to Practice: Designing Learning-based Bidding Strategy	164
5.4.1	DDPGBBS: A Bidding Strategy for Multi-unit Auctions	165
5.4.2	Training and Validation Set-up of DDPGBBS	167
5.4.2.1	Validating the Performance of DDPGBBS	168
5.4.3	Extending DDPGBBS to Operate in Sequential Multi-unit Double Auction of Smart Grids	170
5.4.3.1	Experiments and Results	172
5.5	Summary	174
6	Designing Bidding Strategies via Learning Supply Curve	175
6.1	Introduction	176
6.2	Preliminaries and Mathematical Model	177
6.3	Design of SUPPLYCURVEBBS	178
6.3.1	The Supply Curve Follower (SCF) Module	178
6.3.2	The Bid Generator (BG) Module	179
6.3.3	The Demand Predictor (DP) Module	181
6.4	SUPPLYCURVEBBS in PowerTAC: Experiments and Results	182
6.4.1	Experimental Set-up	183
6.4.2	Results and Discussion	183
6.4.3	SUPPLYCURVEBBS in PowerTAC2021 Tournament	184
6.5	Summary	187

7	Designing Bidding Strategies via Markov Perfect Nash Equilibrium Analysis	188
7.1	Introduction	189
7.2	Preliminaries	191
7.3	The Markov Game Structure	192
7.4	Equilibria of the Markov Game	194
7.4.1	Adequate Supply Case	194
7.4.2	Inadequate Supply Case	200
7.4.3	MPNE-BBS Algorithm	201
7.5	Experimental Evaluation	204
7.5.1	Experimental Setup	204
7.5.2	Results	207
7.5.2.1	Performance Analysis of E-DDPGBBS and MPNE-BBS .	212
7.6	Summary	214
8	Designing Bidding Strategies using MCTS in Continuous Action Space	215
8.1	Introduction	216
8.2	Regression-MCTS (R-MCTS)	219
8.2.1	Selection and Expansion Phase	223
8.2.2	Simulation and Backpropagation Phase	224
8.2.3	Final Selection Phase	225
8.3	Example Walk-through of R-MCTS	225
8.4	Experimental Evaluation	227
8.4.1	Experimental Setup	228
8.4.2	Baseline Strategies	230
8.4.2.1	MCTS-based Strategies	230
8.4.3	Results and Discussion	231
8.5	Summary	235
9	Tariff Generation via Heuristic Method	237
9.1	Introduction	238
9.2	Market Share Analysis	239
9.3	Design of GENTARIFFS-HEUR	244
9.3.1	The Tariff Enhancer (TE) Module	246
9.3.2	The Tariff Designer (TD) Module	247
9.3.3	The Tariff Health Checker (THC) Module	249
9.4	Deploying GENTARIFFS-HEUR in PowerTAC	249
9.5	Experiments and Results	251
9.6	Summary	252

10	Tariff Generation via Contextual-MAB Method	253
10.1	Introduction	254
10.2	Tariff Strategy: A Contextual MAB Approach	255
10.2.1	MDP for the GENTARIFFS-EXP3	255
10.2.1.1	The State Space	256
10.2.1.2	The Action Space	257
10.2.1.3	The Reward Function	258
10.2.1.4	The EXP-3 Algorithm	258
10.3	Deploying GENTARIFFS-EXP3 in PowerTAC	260
10.3.1	Experimental Set-up	261
10.3.2	Results and Discussion	262
10.4	Summary	266
11	Demand Response Model and Method for Peak Reduction	268
11.1	Introduction	269
11.2	Preliminaries and Mathematical Model	271
11.2.1	Modelling the Reduction Probability (RP) Function	272
11.2.2	EXPRESPONSE: The Optimization Problem	276
11.3	Proposed Algorithms for EXPRESPONSE	276
11.3.1	Perfect Information Setting: Known RR	276
11.3.2	Imperfect Information Setting: Unknown RR	279
11.3.3	Experimental Evaluation of MJSUCB–EXPRESPONSE	281
11.4	Adapting MJSUCB–EXPRESPONSE for PowerTAC	282
11.4.1	Experiments and Discussion	285
11.5	Summary	288
12	VidyutVanika in PowerTAC Tournament	289
12.1	Journey of VidyutVanika in the PowerTAC	289
12.2	VidyutVanika: Overview	290
12.3	VidyutVanika21 in the PowerTAC2021 Tournament	291
12.3.1	Design and Overview of VidyutVanika21	293
12.3.2	Wholesale Module of VidyutVanika21	294
12.3.3	Tariff Module of VidyutVanika21	295
12.3.4	Performance Analysis	297
12.4	VidyutVanika22 in the PowerTAC2022 Tournament	304
12.4.1	Design and Overview of VidyutVanika22	305
12.4.2	Wholesale Module of VidyutVanika22	307
12.4.3	Tariff Module of VidyutVanika22	307
12.4.4	Performance Analysis	308
12.5	Guidelines for Autonomous Agent Development for Smart Grids	313
12.6	Summary	315

13 Conclusion and Future Work	316
Bibliography	320

List of Figures

Figure	Page
1.1 Smart Grid System [Image Credit: © [1]]	2
1.2 AI Applications and Impact	3
2.1 Mechanism Design Environment [Image Credit: © [2]]	36
3.1 Reinforcement Learning Framework	60
3.2 Comparison of the update methods of Dynamic Programming (DP), Monte Carlo (MC) and Temporal Difference (TD) learning for state value functions [Image Credit: © [3]]	67
3.3 The DDPG Algorithm Structure Framework [Image Credit: © [4]]	80
3.4 MAB vs. Contextual Bandits	87
4.1 Overview of Smart Grid Technology [Image Credit: © [5]]	99
4.2 Major Elements of the PowerTAC Scenario [Image Credit: © [6]]	102
4.3 Market clearing example: bid 8 and part of ask 6 are the last to clear [Image Credit: © [6]]	108
4.4 Types of Tariffs	110
4.5 Broker's Activities in PowerTAC Markets [Image Credit: © [6]]	112
5.1 Clearing Scenarios for 1 Buyer, 1 Seller and n Units of Items	145
5.2 Relationship between Agent's Actions and Reward	166
5.3 Transitions in of Two-stage MDP of <i>DDPGBBS</i>	166
5.4 Transitions in of Multi-stage MDP of E-DDPGBBS	171
5.5 Average Unit Clearing Price Comparison for five-player game configuration	174
6.1 Overall Average Unit Clearing Price Comparison for in PowerTAC2021 . .	187
7.1 Scenario-1: Wholesale Cost Comparison in 7-Player Games (Set-1: with MISO Buyer)	208

7.2 Scenario-1: Wholesale Cost Comparison in 7-Player Games (Set-2: without MISO Buyer)	209
7.3 Scenario-2: Wholesale Cost Comparison in 7-Player Games (Set-1: with MISO Buyer)	209
7.4 Scenario-2: Wholesale Cost Comparison in 7-Player Games (Set-2: without MISO Buyer)	210
8.1 Tactics to Adjust Risk between Auction Rounds	224
8.2 Pictorial Representation of R-MCTS Displaying all the Phases	226
8.3 Rollouts vs Average Unit Purchase Cost for R-MCTS	231
8.4 Comparing R-MCTS vs MCTS-based Bidding Strategies in Low, Med., High and Ext. Demand Scenarios (((a) vs MCTS-Vanilla (b) vs MCTS-SPW (c) vs SPOT))	233
8.5 Comparing R-MCTS vs PowerTAC Bidding Strategies in Low, Medium, High and Extreme Demand Scenarios (((a) vs ZI (b) vs ZIP (c) vs VV21)) .	234
9.1 GENTARIFFS-HEUR Flowchart	245
9.2 Average Market Share of Brokers in PowerTAC 2021	251
10.1 Market Share Maintained by GENTARIFFS-EXP3 w.r.t Number of Epochs of Training for 3-Player Configuration	265
11.1 Effect of Demand Response	269
11.2 PowerTAC customer's response for FPT and ToU tariffs	274
11.3 DR Probability Function of PowerTAC Customers for (a) The Highest Peak (left), (b) The 2nd Highest Peak (right)	274
11.4 Exponential Probability Function for EXPRESPONSE	275
11.5 Comparing MJSUCB-EXPRESPONSE's regret over 25 iterations with varying batch sizes [budget=5, T=5M]	281
11.6 Comparing MJS-EXPRESPONSE and MJSUCB-EXPRESPONSE with varying budget and number of agents [Iterations = 25, T=5M]	283
12.1 An Abstract Architecture of VidyutVanika	292
12.2 VidyutVanika21: System Architecture	294
12.3 Number of 1st and 2nd ranks of each broker in PowerTAC 2021	300
12.4 Games with Negative Cash in PowerTAC 2021	301
12.5 Average Market Share of Brokers in PowerTAC 2021	301
12.6 Average Market Share of Brokers in Phoenix Games in PowerTAC 2021 .	302
12.7 Accounting Information of Brokers in PowerTAC 2021	303
12.8 VidyutVanika22: System Architecture	306
12.9 Number of 1st and 2nd ranks of each broker in PowerTAC 2022	310
12.10 Games with Negative Cash in PowerTAC 2022	311

12.11Average Market Share of Brokers in PowerTAC 2022	312
12.12Average Market Share of Brokers in Phoenix Games in PowerTAC 2022 . .	314
12.13Accounting Information of Brokers in PowerTAC 2022	314

List of Tables

Table	Page
2.1 Players' Utility Matrix for Prisoner's Dilemma	23
2.2 Individuals' Utility Matrix for Tragedy of Commons	24
2.3 Individuals' Utility Matrix for Battle of Sexes	27
2.4 Example of a Two-player Zero-sum Game	29
5.1 Clearing Conditions and Prices for Buyer's Utility	144
5.2 Clearing Conditions and Prices for Seller's Utility	149
5.3 Value of Scale Factors for Various Cases	160
5.4 Comparison b/w Experimental and theoretical scale factors of buyer Case-1	169
5.5 Comparison b/w Experimental and theoretical scale factors of buyer Case-2	169
5.6 Comparison b/w Experimental and theoretical scale factors of buyer Case-3	170
5.7 Relative Unit Clearing Price Comparison	173
6.1 Mean Wholesale Prices of Brokers for each Player Configuration	184
6.2 Mean Wholesale Prices of Brokers for each Weather Condition	185
6.3 Mean Wholesale Prices each Player Configuration in PowerTAC2021	186
6.4 Mean Wholesale Prices each Weather Condition in PowerTAC2021	186
7.1 Possible Bid Deviations	196
7.2 Scenario-1: Comparison of MPNE-BBS vs E-DDPGBBS in 2-Player Games	209
7.3 Scenario-2: Comparison of MPNE-BBS vs E-DDPGBBS in 2-Player Games	210
7.4 Performance in 2-Player Games (with MISO Buyer)	211
7.5 Performance in 2-Player Games (without MISO Buyer)	212
9.1 2-Player Games Analysis (Utility Values in Millions)	241
9.2 3-Player Games Analysis (Utility Values in Millions)	242
9.3 5-Player Games Analysis (Utility Values in Millions)	242
9.4 7-Player Games Analysis (Utility Values in Millions)	243
10.1 Q-Table for 2-Player Configuration [After 50 Games]	263

10.2 Q-Table for 3-Player Configuration [After 50 Games]	264
10.3 Q-Table for 5-Player Configuration [After 50 Games]	265
11.1 Customer groups detail	284
11.2 Peak usage comparison (usage in MWh)	287
11.3 Capacity Transaction Comparison (Average Penalty)	288
12.1 PowerTAC2021 Leaderboard (Unnormalized)	298
12.2 PowerTAC2021 Leaderboard (Normalized)	299
12.3 Ablation Analysis of VidyutVanika21	304
12.4 PowerTAC2022 Leaderboard (Unnormalized)	309
12.5 PowerTAC2022 Leaderboard (Normalized)	310

Chapter 1

Introduction

“AI-based intelligent agents have the potential to revolutionize industries, optimize processes, and enhance human experiences by combining data, algorithms, and automation.”

– Haleem et. al [7]

We are living in a generation where the world is moving towards efficient utilisation of energy resources, which has given birth to exciting and innovative technologies. Among all the exciting technologies, *smart grid* technology is much more breathtaking and has the capability to change the entire energy spectrum and the way we interact with the electricity grids. Realizing its unrivalled potential, recent years have seen rapid growth in smart grid technology, and some developed nations have already adopted smart grid technology to replace the conventional grid system. The emergence of smart grid technology has opened the door for wide-scale automation in complex decision-making tasks; moreover, manual decision-making for such tasks may result in dwindling efficiency. However, the emergence of *artificial intelligence* has the potential to contribute to the immense automation of

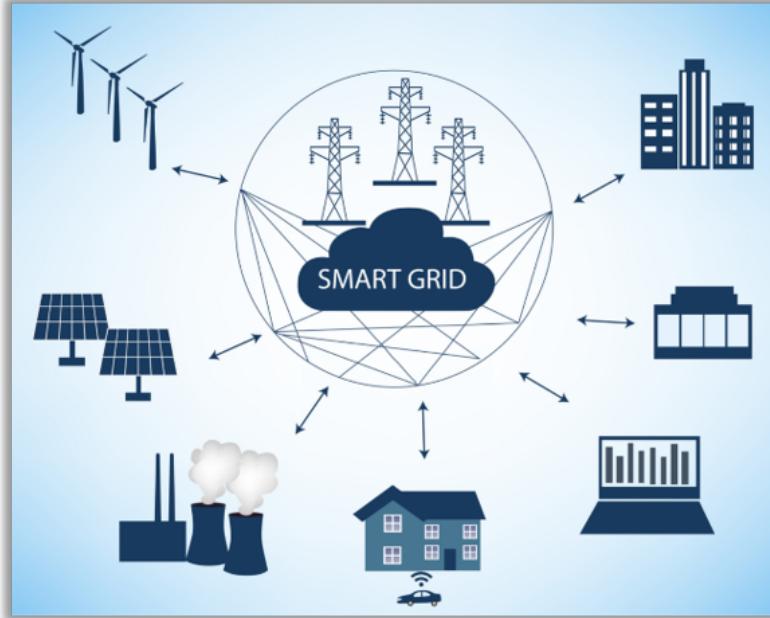
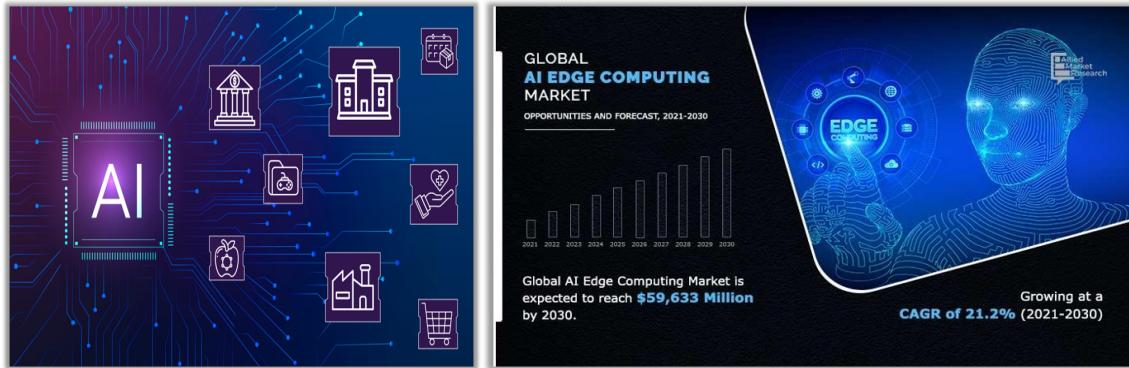


Figure 1.1: Smart Grid System [Image Credit: © [1]]

smart grids. The confluence of two innovative technologies, *smart grids and AI*, is all set to dominate the energy sector in the upcoming years. There are numerous research opportunities to make this confluence highly effective.

The ubiquity of AI, or Artificial Intelligence, is evident in almost every aspect of our lives today. AI has become an integral part of the modern world, from our smartphones and smart homes to the services we use and the products we interact with. One of the main factors contributing to AI's ubiquitous presence is its ability to automate and optimize tasks that humans previously performed. AI-powered systems can analyze vast amounts of data, identify patterns, and make intelligent decisions with speed and accuracy, enabling increased efficiency and productivity across various industries. Furthermore, the increasing availability of data fueled the widespread adoption of AI. The digital age has generated enormous volumes of data, and AI algorithms thrive on data-driven insights. As more

data is collected and processed, AI systems become more accurate and effective, leading to improved predictions, recommendations, and decision-making.



(a) AI Applications [Image Credit: © [8]](b) AI Effects on Global Economy [Image Credit: © [9]]

Figure 1.2: AI Applications and Impact

AI's impact extends beyond consumer applications. Industries such as healthcare, finance, manufacturing, and transportation have embraced AI to optimize operations, improve efficiency, and drive innovation. AI-powered medical diagnostics, fraud detection algorithms, predictive maintenance systems, and autonomous vehicles are just a few examples of how AI is transforming these sectors. As AI continues to evolve and mature, its ubiquitousness will likely expand even further. Advancements in areas such as natural language processing, computer vision, and reinforcement learning may unlock new possibilities and applications. One particularly exciting and emerging application enabled by AI is **automation in smart grid systems**, where **AI-based agents/bots** make decisions on behalf of humans by acting upon the data and deriving patterns out of it.

As per one estimate by Allied Market Research, the global edge computing market is growing at a compound annual growth rate (CAGR) of 21.2% and is expected to reach close to USD 60 million by 2030 [9]. In general, the global AI market was valued at USD 328.34 Billion in 2021 and will likely expand at a compound annual growth rate (CAGR) of 40.2% from 2021 to 2028 [9], and smart grid automation contributes a major share in

it. Smart grid systems involve the trading of electricity in the wholesale market through double auctions. Nord Pool, which runs the leading power market in Europe, showed trades of more than 1000 TWh of volume, with daily transactions exceeding 3 billion USD in Europe alone in 2023 [10]. Among all transactions, close to 60% of the volume was traded using APIs, and the use of AI-based APIs in trading is rapidly increasing even further and is expected to take over the majority of the traded volume in coming years. As AI-based agents are more efficient, even a 5% to 10% improvement in trading can lead overall to profits of millions of USD for a trading company. Clearly, a bidding strategy that can optimize the cost of the trading companies even by a small amount would significantly improve their profits and bring more efficiency. Such progression is possible due to intensive research and computational resources. In general, industries are driving the AI market towards faster, more efficient solutions to maximize revenue. As such, research towards improving certain established performance measures (e.g., the accuracy of a learned model), even marginally, is highly rewarded.

The smart grid system is still emerging with some challenges and potential opportunities to improve. As highlighted, designing agents that can make trading decisions on behalf of humans is exciting. Furthermore, the rising contribution of renewable energy resources in energy generation raises the challenge of efficiently managing fluctuating supply-demand scenarios and grid imbalance situations that may require efficient and robust prediction models. As the smart grid system accommodates more and more players, it requires efficient functioning and planning to manage supply-demand in the grid system and keep the grid stable all the time. All the tasks mentioned above can be tackled with AI and may, in fact, perform much better than their human counterpart.

In the rest of the chapter, we explain the use of smart grids from both economic and customer viewpoints. This is followed by challenges in the smart grid systems that provide research opportunities to design more efficient techniques. Then, we give an overview of all the problems addressed in this work and throw some light on the proposed solutions.

1.1 Motivation to Use Smart Grids

A *smart grid* is an electricity network based on digital technology that is used to supply electricity to consumers via two-way digital communication, which allows for monitoring, analysis, control and communication within the supply chain to help improve efficiency, reduce energy consumption and cost, and maximize the transparency and reliability of the energy supply chain [11]. Smart grid systems aim to overcome the weaknesses of conventional grids by enabling customers to get involved in using smart meters. The utilization of smart grid systems provides more options to effectively manage the use of electricity, both from the suppliers' as well as consumers' perspectives. Below, we describe some motivating factors, from the economic and customers point of view, that smart grids enable.

1.1.1 From Economical Point of View

From an economic standpoint, smart grids have the potential to revolutionize the energy sector by enhancing efficiency, reliability, and sustainability. Here are some key motivations for using smart grids from an economic point of view:

- **Demand Response and Load Management:** Smart grids enable demand response programs, allowing utilities to efficiently manage electricity consumption during peak periods. By providing real-time updates and communication between consumers and grid operators, smart grids enable load shifting and peak load reduction.
- **Energy Efficiency and Cost Savings:** Smart grids empower consumers with real-time information about their energy usage, enabling them to make informed decisions and adjust their consumption patterns accordingly. This increased awareness and control over energy usage can lead to significant energy savings and cost reductions for consumers.

- ***Integration of Renewable Energy Sources:*** Smart grids play a crucial role in integrating renewable energy sources, such as solar and wind, into the existing energy infrastructure. By providing real-time data on energy supply and demand, smart grids facilitate the optimal integration and management of fluctuating renewable energy generation. This integration reduces reliance on fossil fuels, lowers greenhouse gas emissions, and supports the transition to a more sustainable energy system.
- ***Enhanced Grid Reliability and Resilience:*** Smart grids employ advanced monitoring, automation, and self-healing capabilities. These features enable quicker detection and response to grid disturbances, such as power outages or equipment failures. By minimizing downtime and improving grid reliability, smart grids reduce economic losses associated with power interruptions.
- ***Improved Asset Management and Planning:*** Smart grids provide utilities with detailed and real-time data on the performance and condition of grid assets, such as transformers, lines, and substations. This data enables utilities to optimize asset management strategies, identify maintenance needs, predict failures, and extend the lifespan of the equipment. Effective asset management reduces operational costs, avoids unexpected failures, and optimizes capital investments, leading to overall economic benefits for utilities and consumers.

Thus, by modernizing the energy infrastructure and leveraging advanced technologies, smart grids contribute to a more efficient, transparent, reliable, sustainable, and economically viable energy system.

1.1.2 From Customers' Point of View

Smart grids provide several key motivations for customers, offering benefits that enhance their energy experience and empower them to make more informed choices. Here are some motivations for customers to embrace smart grids:

- ***Energy Cost Management:*** Smart grids enable customers to gain better visibility and control over their energy consumption. Real-time information on energy usage and time-of-use pricing and demand response programs allow customers to adjust their energy usage patterns to reduce costs during peak periods, which helps them lower their electricity bills.
- ***Increased Energy Efficiency:*** Smart grids provide customers with detailed insights into their energy usage patterns, helping them identify opportunities for energy savings. Real-time data and smart meters enable customers to track their consumption, identify energy-intensive appliances or behaviours, and make informed decisions to optimize energy efficiency.
- ***Enhanced Reliability and Outage Management:*** Smart grids offer improved reliability and quicker restoration of service during power outages. Customers benefit from reduced downtime, as smart grids facilitate faster fault identification, isolation, and restoration of power. This enhanced reliability ensures a more dependable energy supply, minimizing disruptions and inconveniences.
- ***Integration of Renewable Energy and Distributed Generation:*** Smart grids support the integration of renewable energy sources and distributed generation, such as solar panels or wind turbines. Customers who generate their own renewable energy can feed excess power back into the grid, potentially earning credits or receiving compensation for their contributions.
- ***Improved Customer Service:*** Smart grids enable utilities to offer enhanced customer service experiences. Utilities can provide accurate and timely billing information with remote monitoring and automated meter reading, reducing estimation errors and billing disputes. Customers can access their energy usage data online or through mobile applications, fostering transparency and enabling them to manage their accounts more effectively.

- ***Technological Innovation and Energy Choices:*** Smart grids pave the way for technological innovation and the development of new energy services. Customers can benefit from a range of innovative solutions, such as home automation, energy management systems, and smart appliances that interact with the grid. These advancements provide customers with more options to optimize their energy usage and increase comfort and convenience.

Thus, smart grids empower customers with greater control over their energy usage, promoting energy conservation, sustainability, and a more personalized and responsive energy experience.

1.2 Smart Grid Systems and Its Challenges

The smart grid system is still emerging with some challenges and potential opportunities to improve. Below, we first describe the smart grid system in more detail. Then, we discuss the challenges associated with smart grid systems. After that, we briefly talk about the ways to address those challenges.

A smart grid system refers to an advanced and integrated electricity delivery infrastructure that utilizes modern technologies, communication networks, and intelligent devices to optimize electricity generation, distribution, and consumption. It represents an evolution from traditional power grids by incorporating digital automation, real-time data monitoring, and two-way communication capabilities between various components within the grid. A typical smart grid ecosystem consists of various components like ***wholesale market***, ***tariff market***, ***distribution utility*** and ***electricity brokers***. The wholesale market of a smart grid contains large power-generating companies (GenCos) that act as a primary electricity source. The tariff market incorporates different types of customers, such as consumers, producers, prosumers, and storage. The distribution utility is responsible for electricity distribution from the wholesale market to customers in the tariff market.

The **electricity brokers** are a crucial element of a smart grid that interacts between the wholesale and tariff markets. In smart grids, electricity brokers are the energy distribution companies that procure energy by bidding in the wholesale market auction and selling the procured energy via tariff contracts.

As highlighted previously, some challenges and potential opportunities exist to improve smart grid systems. The first challenge is the rising contribution of renewable energy resources in energy generation, which raises the challenge of efficiently managing fluctuating supply-demand scenarios and grid imbalance situations. Secondly, the problem occurs when there is a sudden surge of consumption (i.e., during heat waves) and the demand goes beyond the normal working range of supply, leading to demand peaks. Peak demands lead to added load on GenCos to supply additional energy through fast ramping mechanisms to fulfil the energy requirement, leading to higher costs for distribution companies. Lastly, from the electricity broker's perspective, managing the supply-demand balance within its portfolio in a smart grid ecosystem is an existing challenge.

The solution to the above challenges can be designed from an electricity broker's point of view. As stated, the broker plays an essential role in the smart grid's operations and is in charge of buying and selling electricity in the smart grid system. A broker must be equipped with an effective demand predictor to predict the customers' and market demand to determine the required electricity to be purchased. A broker has to design an **efficient bidding strategy** to purchase electricity from the wholesale market by taking part in **day-ahead periodic double auctions** with the aim of minimizing purchase cost. After procuring electricity, the broker needs to sell it to retail customers in the tariff market by publishing lucrative yet profitable **tariff contracts** with the *aim of maximizing profits* through electricity sales. However, a broker may get penalized if it is found to be contributing to **demand peaks** because of the sudden surge in consumption by some of the customers under its portfolio. Thus, it needs to effectively incentivize customers to shift their non-essential usages to non-peak timeslots by utilizing techniques like **demand**

response. A detailed discussion of a broker's activities in a smart grid system is presented in Section 4.3.

Any solutions designed for a smart grid system must be validated to test its efficacy. However, validating the novel solution techniques directly on real-world smart systems is impractical and may lead to catastrophe if the solution malfunctions. Thus, to provide a platform to enable smart grid research, *Power Trading Agent Competition (Power-TAC)* designed an efficient and close-to-real-world smart grid simulator [6]. PowerTAC mimics all the crucial elements of a smart grid system and acts as a test bed to validate novel strategies.

1.3 Problems Addressed

In this section, we describe the problems addressed in the thesis in detail. We specifically solve problems related to two markets: the wholesale and tariff markets. The proposed solution indirectly affects the balancing markets as well.

In the wholesale market, the broker needs to design an efficient bidding strategy to purchase electricity from the wholesale market by participating in day-ahead periodic double auctions (PDAs). A day-ahead auction enables the sale of electricity 24 hours prior to the delivery timeslot; thus, the bidding strategy requires planning and consideration of current and future auctions. If the broker fails to procure the necessary electricity to fulfil its customer demand after all auctions, it must buy the remaining electricity from the balancing market at expensive rates; thus, a bidding strategy should aim to procure all the required electricity from the wholesale market only.

After procuring electricity from the wholesale market, the broker needs to sell it to customers in the tariff market by publishing tariff contracts. Tariffs should be such that they attract customers yet also be profitable to the broker. By publishing tariff contracts in the tariff market, a broker creates a portfolio of customers, which may include various types of customers like households, offices, storage, production, etc. The tariff contract details

are available to all the brokers in the market; thus, the opponent brokers may update their own tariffs to offer a better tariff than our broker and take away our customers. So, a broker needs to update its tariff periodically to remain competitive in the market.

Finally, the peak demands problem is one of the most prevalent problems in smart grids, where a broker gets heavily penalized if found guilty of contributing to peak demands. These penalties are proportional to the broker's market share. Thus, a broker needs to find ways to distribute electricity use proportionally throughout the day to mitigate peak demand scenarios.

Below, we present a brief overview of all the proposed approaches to tackle the above-mentioned problems.

1.4 Proposed Approaches

In this section, we give a brief overview of all the strategies proposed in this work. We divide this section into two segments, *Part A: The Wholesale Market*, and *Part B: The Tariff Market*. In Part A, we overview the proposed wholesale strategies. Part B is further divided into two parts. In Part B₁, we provide a brief overview of tariff contract generation strategies, whereas in Part B₂, we present a strategy to achieve peak reduction in smart grids.

1.4.1 Part A: In the Wholesale Market of a Smart Grid

As described earlier, the broker needs to design an efficient bidding strategy to purchase electricity from the wholesale market by participating in day-ahead periodic double auctions (PDAs). Unlike the strategies presented in the literature [12, 13], which may not be suitable for real-time auctions, we present various novel bidding strategies, most of which are real-time and easy to deploy.

1.4.1.1 Bidding Strategy via Equilibrium Theory

In this work, we apply game theory concepts to carry out a *Bayesian Nash Equilibrium* analysis for single-buyer single-seller multi-unit double auctions.

Our approach is motivated by the literature where an equilibrium strategy often derived by multiplying the true valuation of a player by a scalar value [14], which is known as *scale-based bidding strategy*. We focus on scaling-based bidding strategies for the following reasons: - (i) they are natural when the brokers need to submit bids in real-time as the broker would want to scale-down its valuation while submitting the bid to become profitable. (ii) Scale-based strategies are easier to deploy than other techniques like solving differential equations or fictitious play formulations, as, during the inference time (or run-time), the bid is just a scalar multiplication of the scaling factor and the broker's valuation. Hence, the primary reason to use scale-based strategies is their simplicity during the execution, as an agent needs to bid in real-time in settings like smart grids.

After presenting a formal analysis for single-buyer, single-seller double auctions, we extend the idea to be useful in practical scenarios involving more than one buyer/seller in real-world PDAs. We propose a *learning-based strategy* inspired by the techniques in *reinforcement learning* to determine the scale factors for buyer/seller for the scale-based bidding strategy. During the execution time, such strategies would determine the scaling factor that would get multiplied by the broker's valuation to place the bid.

1.4.1.2 Bidding Strategy via Learning Supply Curve

In this work, we take a different approach and utilize the knowledge about the supplier in the wholesale market. Particularly, we reconstruct the cost curve (or supply curve) of the prominent power-generating company (GenCo) in the market. The bidding strategy aims to utilize the knowledge of this *supply curve* to make bids in the auctions. It aims to reduce procurement costs by identifying the lowest ask for each auction using the supply curve, which is then used to generate suitable bids. The lowest ask acts as an upper limit on

the generated bids. By placing such bids, the broker attempts to procure electricity from fellow brokers who sometimes sell the extra electricity in the market at minimal prices. However, such brokers do not exhibit any selling pattern and the GenCo acts as the last resort to procure the remaining electricity.

1.4.1.3 Bidding Strategy via Markov Perfect Nash Equilibrium

In this work, we propose *Markov Perfect Nash Equilibrium (MPNE)* policies for a competitive setting where multiple prosumers (buyers who can sell items) who have knowledge about GenCo's supply curve compete to procure commodities. Building on these MPNE policies, we develop an algorithm for scenarios where the buyers do not have knowledge about the supply curve and need to re-construct an approximate composite supply curve using past auction information. Based on the approximate supply curve, the algorithm determines a range of bid prices to generate suitable bids.

1.4.1.4 Bidding Strategy using MCTS in Continuous Action Spaces

In this work, we employ *Monte Carlo Tree Search (MCTS)*, a state-of-the-art online planning algorithm for sequential decision-making, to design an effective bidding strategy for PDAs. We propose a bidding strategy for a continuous action space of bid prices that leverages information obtained from explored actions to enhance the understanding of the larger action set within the continuous domain to place bids in the auctions, thus generalizing the information about action quality between a wider action space for faster learning. We build the Monte Carlo tree for the proposed strategy by performing a sufficient number of rollouts, which are then used to generate suitable bids.

1.4.2 Part B: In the Tariff Market of a Smart Grid

In the tariff market, the broker needs to sell electricity via tariff contracts. The broker also needs to mitigate peak demand scenarios. We divide this section into two parts: Part

B_1 presents various tariff contract generation strategies, while Part B_2 discusses the peak reduction technique.

1.4.3 Part B_1 : Tariff Contract Generation

The electricity procured from the wholesale market needs to be sold to the customers in the retail market, for which a broker has to publish tariff contracts. Below, we provide a brief overview of proposed tariff-generation strategies.

1.4.3.1 Tariff Contract Generation via Heuristic Method

For this strategy, we first show that acquiring all the customers in the tariff market (having 100% market share) is not an optimal strategy to maximize the profit as it may lead to *huge peak demand penalties*. Using a *game theoretical analysis*, we determine an *optimal market share* for the broker depending on the number of opponents in the game, which leads to higher profits in the games. After that, we design a *heuristic-based strategy* that achieves and maintains the optimal market share suggested by the game theory analysis. The proposed strategy also aims to create a balanced portfolio of customers that includes different types of customers (consumption, production and storage) to contribute to avoiding peak-demand penalties.

1.4.3.2 Tariff Contract Generation via Contextual-MAB Method

In our second tariff contract generation strategy, we take our previous heuristic-based strategy as a base and aim to replace the heuristics with the *learning-based method* to achieve and maintain the optimal market share suggested by the game theory analysis. To make the strategy learn in the games, we utilize *Multi-armed Bandit (MAB)* literature to design the tariff generation strategy. This strategy, too, uses the game theory analysis to determine an optimal market share in games and adapts itself to achieve and maintain that market share during the game.

1.4.4 Part B₂: Peak Reduction via Demand Response

Finally, we address the problem concerning peak demands. The above-presented strategies are one way to target an appropriate market share to avoid paying heavy penalties. However, another way to tackle the problem is to use *demand response (DR)* techniques to offer *incentives* to customers to help reduce peak demand. Below, we present our proposed DR technique for peak reduction.

1.4.4.1 Demand Response Model and Method for Peak Reduction

This work studies the effect of DR incentives on the probabilities of customers shifting their usages away from peak timeslots. We first model a function depicting the reduction probability (RP) of a customer reducing its load given various discounts, which turns out to be an exponential function of discounts. Each customer has a different RP, which is parametrized by the rate of reduction (RR). When the RRs of the customers are known, we provide an optimal algorithm that outputs the discounts to each customer by maximizing the expected reduction under a budget constraint. When RRs are unknown, we propose a Multi-Armed Bandit (MAB) based online algorithm to learn RRs. Experimentally, we show that it exhibits sublinear regret. Then, we extend the method to be useful in practical scenarios involving thousands of customers and showcase the efficacy of the method in reducing peak demand penalties.

1.5 Outline of the Work

In this section, we provide a brief outline of this thesis. As mentioned, the thesis is divided into Part A: The Wholesale Market - Designing Bidding Strategies in Wholesale Market, and Part B: The Tariff Market. Part B is further divided into Part B₁: Designing Tariff Generation Strategies in Tariff Market, and Part B₂: Designing Tariff-based Demand

Response for Peak Reduction in Smart Grid. Before that, we covered fundamental concepts utilized in this work.

- **Chapter 2:** In this chapter, we include the fundamental concepts and definitions of game theory and mechanism design.
- **Chapter 3:** Here, we include a detailed discussion about the reinforcement learning (RL) fundamentals and various RL algorithms.
- **Chapter 4:** This chapter provides a comprehensive discussion about the components of a smart grid, the PowerTAC simulator, and the broker's activities in the PowerTAC. It also includes an extensive literature survey of the strategies designed for the smart grid systems and PowerTAC, in particular.

Part A: The Wholesale Market - Designing Bidding Strategies

- **Chapter 5:** Here, we present the scaling-based bidding strategy inspired by equilibrium theory for multi-unit double auctions.
- **Chapter 6:** In this chapter, we present the supply curve-based bidding strategy where we place bids via learning the supply curve of the prominent power-generating company in the wholesale market.
- **Chapter 7:** Here, we propose Markov perfect Nash equilibrium (MPNE) policies when the prosumers (buyers who can sell) with knowledge of the composite supply curve participate in PDAs, which is then utilised to devise an algorithm for the real-world scenario where the supply curve information is not known.
- **Chapter 8:** In this chapter, we propose an MCTS-based bidding strategy that operates in the continuous space of bid prices and places bids in the market by utilising knowledge of previously explored actions in MCTS.

Part B₁: The Tariff Market - Tariff Contract Generation Strategies

- **Chapter 9:** Here, we present a heuristic-based tariff contract generation that aims to achieve and maintain an equilibrium market share suggested by a game theory analysis.
- **Chapter 10:** In this chapter, we replace the heuristics with the contextual multi-armed bandit-based learning strategy to generate tariff contracts.

Part B₂: The Tariff Market - Demand Response (DR) Strategies for Peak Reduction

- **Chapter 11:** Here, we present the demand response-based tariff strategy that models the DR behaviour of the customers and utilizes it to provide incentives to achieve peak reduction in smart grids.
- **Chapter 12:** Finally, we present the details of our broker agent, *Vidyut Vanika*, deployed in the PowerTAC tournaments in the year 2021 and 2022. We provide an extensive analysis of the above-presented strategies in the PowerTAC tournaments.

Finally, we present a conclusion of the thesis and a direction for future work in Chapter 13.

Chapter 2

Game Theory and Mechanism Design

“The best for the group comes when everyone in the group does what’s best for himself AND the group.”

– Prof. John Nash

Game theory and **mechanism design** are two intertwined fields that delve into strategic interactions and the design of systems to achieve desired outcomes. Game theory analyzes decision-making in situations where the actions of one participant affect the outcomes of others, while mechanism design focuses on constructing rules and mechanisms that incentivize desired behaviour and outcomes. Game theory provides a framework for players to select strategies to **maximize their utility**, where the outcome of one person’s choice depends on the choices made by others. Mechanism design, on the other hand, seeks to design systems or mechanisms that **elicit desirable behaviours** or outcomes from self-interested participants.

The interplay between game theory and mechanism design is crucial. Game theory provides a foundation for understanding strategic interactions and predicting outcomes, while mechanism design harnesses this understanding to engineer systems that achieve desired outcomes in the face of self-interested agents. Together, they offer valuable insights into various domains, including economics, auctions, voting systems, market design, and resource allocation.

2.1 Game Theory: An Overview

Game theory (GT) is a branch of applied mathematics used to analyze situations involving **intelligent, rational players**. It is a fascinating field of study that explores the strategic interactions between individuals, businesses, or nations. It provides a framework for analyzing decision-making in situations where the outcome of one person's choice depends on the choices made by others. More formally, "*game theory is the study of mathematical models of conflict and cooperation between intelligent, rational decision-makers*". At its core, game theory seeks to understand how rational actors behave when faced with **competitive** or **cooperative** situations. It examines the incentives, strategies, and potential outcomes that arise when multiple players are involved.

One of the fundamental concepts in game theory is **the game** itself, which represents a formalized model of strategic interaction. Games can take various forms, from simple two-player scenarios to complex multi-player settings. Each player aims to **maximize their own payoff or utility**, considering the actions and strategies of others. Game theory provides tools for analyzing situations in which players make interdependent decisions. This interdependence between players' decisions causes each player in a game to consider other players' possible decisions while formulating its strategy. A **strategy** is an algorithm or a set of rules, following which each player decides its action in each situation (state). It

helps shed light on the dynamics of conflicts and cooperation, highlighting the trade-offs and interdependencies involved.

Nash Equilibrium (NE) is one of the most important solution concepts in game theory, which defines strategy profiles where no player is better off by unilaterally deviating from its equilibrium strategy. However, there are games where at least one player contains private information about the game that is not accessible to the other players, leading to incomplete information games. Incomplete information games are more realistic and are studied using **Bayesian Nash Equilibrium (BNE)**, which is defined later in the chapter.

Game theory has applications in diverse fields, including economics, politics, biology, and even everyday situations. It helps explain phenomena like pricing wars between companies, negotiations between countries, or even the dynamics of personal relationships.

2.1.1 What is Game Theory? Definitions and Examples

Game theory has been defined formally by various authors; below are some of the definitions. The first definition of game theory was proposed by John von Neumann in 1944, says “*Game theory is concerned with the analysis of conflict and cooperation in decision-making between rational individuals*”. John von Neumann and Oskar Morgenstern’s book, “*Theory of Games and Economic Behavior*”, is considered the foundation of modern game theory. In 1950, John Nash defined game theory as “*Game theory is a mathematical tool to model and analyze situations involving conflict and cooperation between rational decision-makers*”. John Nash, an American mathematician and Nobel laureate, made significant contributions to the field of game theory, particularly with his concept of Nash equilibrium, which explores stable outcomes in games.

As discussed before, **the game** is the fundamental concept in game theory. A game is a formalized model of strategic interaction that involves skill, chance and endurance. Below are the elements of a game:

- **Players:** The agents playing the game, denoted by $N = \{1, 2, \dots, n\}$
- **States:** Represents all possible situations in the game, denoted by S
- **Actions:** A set of decisions that agents can take, which change the state of the game, denoted by A
- **Beliefs:** The knowledge of the states and actions, denoted by p_i
- **Outcome:** Set of outcomes for a player's actions, denoted by X , in particular, the payoff for each player
- **Utility:** Payoff that each player derives from the outcome, denoted by u_i

We make a few assumptions; every player acts rationally so as to maximize their own payoff (note that the payoff may not always be the profits), and information about the game is **common knowledge**. Common knowledge says that every player knows it (it is mutual knowledge), and all of the players know that all other players know it, and all other players know that all other players know that all other players know it, and so on.

A game can be represented as an **Extensive Form Game** or a **Strategic Form Game**¹. An extensive form game represents a game as a tree, detailing the sequential structure of decisions, the players' possible actions at each stage, the information available to them, and the payoffs at each outcome. It is well-suited for analyzing games with **sequential moves** and imperfect information.

However, the extensive form games have limitations due to their tree-like structure, as representing and analyzing large games becomes cumbersome due to the exponential growth of the tree structure. Solving extensive form games (e.g., finding an equilibrium solution) can be computationally expensive. Thus, it is mainly used to analyze the games and observe the dynamic interactions among the players.

¹All the definitions in this chapter are taken from the book “Game Theory and Mechanism Design” by Prof. Narahari [15]

The limitations of extensive form games led to the strategic form representation, which simplifies games by summarizing them in a matrix. The strategic form focuses on overall strategies and their payoffs, making it easier to study Nash equilibria and analyze complex scenarios, especially in ***simultaneous-move*** or mixed-strategy games

Definition 2.1.1: Strategic Form Game [16]

A strategic form game Γ is represented as a tuple $\langle N, (S_i)_{i \in N}, (u_i)_{i \in N} \rangle$ where,

- $N = \{1, 2, \dots, n\}$ is the set of players
- S_1, S_2, \dots, S_n are the strategies of the players
- $u_i : S_1 \times S_2 \times \dots \times S_n \rightarrow \mathbb{R}$ for $i \in N$ are the utility functions

Below, we present an example of ***prisoner's dilemma*** to explain the above concepts in detail.

Example 2.1.1: Prisoner's Dilemma

This game has two prisoners $N = \{A, B\}$. The prosecutors have no evidence to convict them. However, the prosecutors question each of the prisoners separately to obtain a confession. The prisoners cannot communicate with each other and are offered the following choices,

- If both confess, they will each receive 5-years imprisonment
- If prisoner A confesses and prisoner B defects, then prisoner A gets 1 year imprisonment, and B gets 10 years
- If prisoner B confesses and prisoner A defects, then prisoner B gets 1 year imprisonment, and A gets 10 years
- If both defect, they get 2 years imprisonment each

Table 2.1: Players' Utility Matrix for Prisoner's Dilemma

<i>A</i>	<i>B</i>	Defect (D)	Confess (C)
Defect (D)		(-2, -2)	(-10, -1)
Confess (C)		(-1, -10)	(-5, -5)

We obtain the payoff matrix for the prisoners (*A* and *B*) as shown in Table 2.1. The row player is *A*, and the first value in each cell corresponds to its utility, while *B* is the column player, and the second value in each cell corresponds to its utility. *D* and *C* denote the prisoner's choice to defect and confess, respectively. The representation in Table 2.1 is also called the *matrix form* of the game. Similar to the prisoner's dilemma, many other situations can be analyzed by modelling them as a strategic form game. Below is one such example of *tragedy of commons* game.

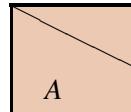
Given the game of Prisoner's Dilemma, what should the rational and intelligent prisoners choose, and hence, what would the outcome be? What is the optimal choice for both prisoners? For the Tragedy of Commons game, would the individuals be interested in keeping the city clean? These questions lead us to the various equilibrium concepts in game theory, which are discussed in Section 2.1.2.

Example 2.1.2: Tragedy of Commons

This game has two individuals $N = \{A, B\}$. The individuals aim to keep their surroundings clean. Below are the payoffs offered for their choices,

- If both players keep their surroundings clean, they BOTH will receive a utility of 50 (so, total utility 100 to each player)
- A player incurs a utility of -60 in his efforts to keep surrounding clean

Table 2.2: Individuals' Utility Matrix for Tragedy of Commons

	Clean (C)	Dirty (D)
Clean (C)	(40, 40)	(-10, 50)
Dirty (D)	(50, -10)	(0, 0)

2.1.2 Preliminaries and Properties of a Game

Given a game Γ , players may choose different strategies based on the utility. Below, we define some important definitions. We define two different equilibrium strategies for the players, *Dominant Strategy Equilibrium* and *Nash Equilibrium*. Below, we first define a *Strategy* and its types,

Definition 2.1.2: Strategy

A *strategy* is a complete contingent plan explaining how a player will act in each state. In other words, it is an algorithm or rule by which each player chooses an action.

Definition 2.1.3: Pure Strategy [15]

A strategy is a pure strategy if, for each state, the action is chosen in a deterministic way.

Definition 2.1.4: Mixed Strategy [15]

For player i , its mixed strategy σ_i is a probability distribution over the strategy set S_i , i.e., $\sigma_i(s_i), s_i \in S_i$ indicates the probability with which player i plays s_i .

A strategy maximizing the utility of a player irrespective of the strategy of the other players is the dominant strategy. A dominant strategy equilibrium is the dominant strategy profile of all players. Any rational, intelligent player chooses to play the dominant strategy if such an equilibrium exists. Below, we define two types of dominant strategy equilibrium formally, namely ***Weakly Dominant Strategy Equilibrium*** and ***Strongly Dominant Strategy Equilibrium***,

Definition 2.1.5: Weakly Dominant Strategy Equilibrium [15]

Given a game Γ , a strategy profile (s_1^*, \dots, s_n^*) called a ***weakly dominant strategy equilibrium***, if $\forall i \in N$, the strategy s_i^* is a dominant strategy,

$$u_i(s_i^*, s_{-i}) \geq u_i(s_i, s_{-i}), \forall s_i \in S_i, s_i \neq s_i^* \text{ and } s_{-i} \in S_{-i}$$

while it is $u_i(s_i^*, s_{-i}) > u_i(s_i, s_{-i})$ for some $s_{-i} \in S_{-i}$.

Definition 2.1.6: Strongly Dominant Strategy Equilibrium [15]

Given a game Γ , a strategy profile (s_1^*, \dots, s_n^*) called a ***strongly dominant strategy equilibrium***, if $\forall i \in N$, the strategy s_i^* is a strongly dominant strategy,

$$u_i(s_i^*, s_{-i}) > u_i(s_i, s_{-i}), \forall s_i \in S_i, s_i \neq s_i^* \text{ and } s_{-i} \in S_{-i}$$

Let us see a few examples to understand the above definitions. Recall Example 2.1.1; below, we attempt to find dominant strategy equilibria for these games.

Example 2.1.3: Finding Equilibrium of Prisoner's Dilemma

This game has two prisoners $N = \{A, B\}$. The prisoners cannot communicate with each other and are offered the following choices: Defect (D) or Confess (C). Depending on the other player's choice, each player would get some utility, as shown in Table 2.1.

From the Table 2.1, we can write following,

$$u_A(C, D) > u_A(D, D) \text{ and } u_A(C, C) > u_A(D, C)$$

$$u_B(C, D) > u_B(D, D) \text{ and } u_B(C, C) > u_B(D, C)$$

Thus, (*Confess (C), Confess (C)*) is the strongly dominant strategy equilibrium for prisoner's dilemma. Intuitively, choosing to *Confess* is the best response for each player, irrespective of what the player is choosing.

Similarly, for Example 2.1.1, (*Dirty (D), Dirty (D)*) is the strongly dominant strategy equilibrium.

Example 2.1.4: Finding Equilibrium of Tragedy of Commons

This game has two individuals $N = \{A, B\}$. The individuals aim to keep their surroundings clean and are offered the following choices: Clean (C) or Dirty (D). Depending on the other player's choice, each player would get some utility, as shown in Table 2.2.

From the Table 2.2, we can write following,

$$u_A(D, C) > u_A(C, C) \text{ and } u_A(D, D) > u_A(C, D)$$

$$u_B(D, C) > u_B(C, C) \text{ and } u_B(D, D) > u_B(C, D)$$

Thus, (*Dirty (D), Dirty (D)*) is the strongly dominant strategy equilibrium for the tragedy of commons. Intuitively, choosing to *Dirty* is the best response for each player, irrespective of what the player is choosing. This explains why even though everyone wants their surroundings to be clean, they remain dirty.

However, such a strategy *may not always exist*. Let us example below.

Table 2.3: Individuals' Utility Matrix for Battle of Sexes

<i>A</i>	<i>B</i>	Football (F)	Opera (O)
Football (F)		(2, 1)	(0.5, 0.5)
Opera (O)		(0, 0)	(1, 2)

Example 2.1.5: Battle of Sexes Game

This game has two individuals $N = \{A, B\}$. Both individuals have conflicting preferences but want to coordinate their actions. Below are the payoffs offered for their choices,

- If both players choose to go to watch football, they will receive a utility of 2 and 1, respectively, as Player A prefers to watch football more than Player B .
- If both players choose to go to watch opera, they will receive a utility of 1 and 2, respectively, as Player B prefers to watch opera more than Player A .
- If Player A chooses to watch football and Player B chooses to watch opera, they both receive a utility of 0.5 and 0.5, respectively.
- If Player A chooses to watch opera and Player B chooses to watch football, they both receive a utility of 0.

Let us attempt to find the dominant strategy for the game of battle of sexes.

Example 2.1.6: Finding Dominant Strategy for Battle of Sexes Game

From the Table 2.3, we can write following,

$$u_A(F, F) > u_A(O, F) \text{ and } u_A(F, O) < u_A(O, O)$$

Thus, there is no dominant strategy equilibrium for the game of battle of sexes.

As shown in the above example, the dominant strategy equilibrium may not always exist. Now, there are two directions: (i) restrict our attention to a simpler class of games, i.e., two-player zero-sum games, or (ii) introduce a weaker notion of equilibrium. Let us first see the two-player zero-sum games and whether equilibrium always exists in such games.

Definition 2.1.7: Two-player Zero-sum Game [15]

A two-player zero-sum game $\Gamma = \langle N, S_1, S_2, u_1, u_2 \rangle$ is a game where,

- $N = \{1, 2\}$ is the set of players
- S_1, S_2 are the strategies of the players
- $u_1 : S_1 \times S_2 \rightarrow \mathbb{R}$ for $i \in N$ are the utility functions, where, $u_1 + u_2 = [0]$, meaning $u_2 = -u_1$.

Thus, in a two-player zero-sum game, only the first player's utility matrix is enough to know the utility matrix of the second player. This game setting is also called **matrix games**. Table 2.4 shows an example of a two-player zero-sum game. Such game formulation is widely adopted, and we use a similar formulation to analyze our game setting in Chapter 9.

Two-player zero-sum games simplify the game; however, in such games, too, a **dominant strategy does not always exist**, as shown in the example given in Table 2.4.

Table 2.4: Example of a Two-player Zero-sum Game

<i>A</i>	<i>B</i>	Left (L)	Middle (M)	Right (R)
Top (T)		1	2	2
Middle (M)		-3	-1	-2
Bottom (B)		-2	0	1

Example 2.1.7: Finding Dominant Strategy for the Game in Table 2.4

From the Table 2.4, we can write following,

$$u_A(T, L) > u_A(M, L) \text{ and } u_A(T, L) > u_A(B, L)$$

$$u_A(T, M) > u_A(M, M) \text{ and } u_A(T, M) > u_A(B, M)$$

$$u_A(T, R) > u_A(M, R) \text{ and } u_A(T, R) > u_A(B, R)$$

Thus, playing Top (*T*) is the strongly dominant strategy for player *A*. However, player *B* does not have any dominant strategy (no column that dominates another column). Thus, this game does not have any dominant strategy equilibrium.

Hence, Nash proposed a weaker notion of equilibrium referred to as the **Nash equilibrium**. A strategy profile is Nash equilibrium if every player following the strategy maximizes their utility, given that every other player also follows the given strategy. In the following definitions, we consider n players having a strategy profile (s_1, \dots, s_n) where s_{-i} denotes the players' strategy tuple except player i . Below, we define two types of Nash equilibrium based on the resultant strategy profile, namely, **Pure Strategy Nash Equilibrium** and **Mixed Strategy Nash Equilibrium (MSNE)**,

Definition 2.1.8: Pure Strategy Nash Equilibrium [16]

Given a game Γ , a strategy profile (s_1^*, \dots, s_n^*) is called a ***pure strategy Nash equilibrium***, if $\forall i \in N$, the strategy s_i^* satisfies,

$$u_i(s_i^*, s_{-i}^*) \geq u_i(s_i, s_{-i}^*), \forall s_i \in S_i$$

This means that unilateral deviation from any player would not help him to achieve a better utility.

Definition 2.1.9: Mixed Strategy Nash Equilibrium (MSNE) [15]

Given a N player game $\Gamma = < N, (S_i), (u_i) >$, a mixed strategy profile $(\sigma_1^*, \dots, \sigma_n^*)$ is called a ***mixed strategy Nash equilibrium*** if, $\forall i \in N$, $u_i(\sigma_i^*, \sigma_{-i}^*) \geq u_i(\sigma_i, \sigma_{-i}^*)$, $\forall \sigma_i \in \Delta(S_i)$. σ_{-i}^* denotes mixed strategies of all players except i .

Example 2.1.8: Finding Nash Equilibrium of Battle of Sexes Game

This game has two individuals $N = \{A, B\}$. Both individuals have conflicting preferences but want to coordinate their actions. From the Table 2.3, we can write following,

$$u_A(F, F) > u_A(O, F) \text{ and } u_B(F, F) > u_B(F, O)$$

$$u_A(O, O) > u_A(F, O) \text{ and } u_B(O, O) > u_B(O, F)$$

Thus, **(Football (F), Football (F))** and **(Opera (O), Opera (O))** are Nash equilibrium for the battle of sexes. Intuitively, both players would coordinate with each other.

The aforementioned definition of Nash equilibrium assumes complete information, implying that all players have full knowledge of the game, including each other's valuations

and utilities. However, in many practical setups, each player possesses ***private information***, such as their true type or valuation of the commodity. This introduces the complexity of incomplete information into the game. An advanced solution concept known as ***Bayesian Nash equilibrium*** is available to effectively analyze such scenarios. This equilibrium framework accounts for the fact that players must make strategic decisions based on their private information while considering the probability distributions of other players' private information. Below, we first define a strategic form game with incomplete information, followed by the definition of Bayesian Nash equilibrium.

Definition 2.1.10: Strategic Form Game with Incomplete Information [15]

A strategic form game with incomplete information or ***Bayesian Game*** is defined as a tuple $\Gamma = \langle N, (S_i), (\Theta_i), (u_i), (p_i) \rangle$ where,

- $\{N = 1, 2, \dots, n\}$ is the set of players.
- S_1, S_2, \dots, S_n are the strategies of the players
- $\Theta_1, \Theta_2, \dots, \Theta_n$ are the sets of types of the players
- $u_i : S_1 \times S_2 \times \dots \times S_n \times \Theta_1 \times \Theta_2 \times \dots \times \Theta_n \rightarrow \mathbb{R}$ for $i \in N$ are the utility functions
- The belief function p_i is a mapping from Θ_i into $\Delta(\Theta_{-i})$, the set of probability distributions over Θ_{-i} . That is, for any possible type $\theta_i \in \Theta_i$, p_i specifies a probability distribution $p_i(\cdot | \theta_i)$ over the set Θ_{-i} representing player i 's beliefs about the types of the other players if his own type were θ_i .

Definition 2.1.11: Bayesian Nash equilibrium [15]

A ***Bayesian Nash equilibrium*** in a Bayesian game $\Gamma = \langle N, (S_i), (\Theta_i), (u_i), p \rangle$ (game with incomplete information) is defined as, a profile of strategies $(s_1^*, s_2^*, \dots, s_n^*)$

is a Bayesian Nash Equilibrium, if $\forall i \in N; \forall s_i : \Theta_i \rightarrow S_i; \forall \theta_i \in \Theta_i$,

$$u_i((s_i^*, s_{-i}^*) \mid \theta_i) \geq u_i((s_i, s_{-i}^*) \mid \theta_i)$$

That is, $\forall i \in N; \forall a_i \in S_i; \forall \theta_i \in \Theta_i$,

$$\mathbb{E}_{\theta_{-i}}[u_i(\theta_i, \theta_{-i}, s_i^*(\theta_i), s_{-i}^*(\theta_{-i}))] \geq \mathbb{E}_{\theta_{-i}}[u_i(\theta_i, \theta_{-i}, a_i, s_{-i}^*(\theta_{-i}))]$$

where, N is a set of players, θ_i and $u_i(\cdot)$ are true type and utility of player i , respectively. Θ_i denotes the set of private values of agent i , $\forall i \in N$.

Now, recall Example 2.1 and Example 2.2. In the prisoner's dilemma game, note that choosing D is strongly dominated by C for A since,

$$u_A(C, D) > u_A(D, D) \text{ and}$$

$$u_A(C, C) > u_A(D, C)$$

Similarly for B it is,

$$u_B(D, C) > u_B(D, D) \text{ and}$$

$$u_B(C, C) > u_B(C, D)$$

Thus (C, C) is a strongly dominant strategy. Further, it is also the unique Nash equilibrium. Although (C, C) is the natural prediction, (D, D) is the best outcome jointly for the players. Hence, the dominant strategy outcome does not maximize the players' overall welfare or the sum of utilities.

For the Tragedy of Commons game, keeping the city dirty is the dominant strategy for individuals A and B .

For individual A ,

$$u_A(D, C) > u_A(C, C) \text{ and}$$

$$u_A(D, D) > u_A(C, D)$$

For individual B ,

$$u_B(C, D) > u_B(C, C) \text{ and}$$

$$u_B(D, D) > u_B(D, C)$$

So, in the Tragedy of Commons game, (D, D) is a strongly dominant strategy as well as Nash equilibrium. Thus, it answers why our cities are dirty even though everyone wants them to be clean.

Equipped with the essential game-theoretic tools that help to understand the strategic decision-making of rational actors in various interactive situations, we try to model real-world situations similarly to obtain desirable outcomes. In many real-world scenarios, decisions are made considering the interests of a group of people, especially in public decision-making and resources/task allocation in any organization. In such scenarios, **mechanism design** provides a way to elicit the preferences of individuals and allocate the resources. Mechanism design builds upon game theory's insights and employs its analytical tools to design rules, protocols, and incentive structures that shape participants' behaviour and achieve desired outcomes. By leveraging the understanding of strategic interactions gained from game theory, mechanism design seeks to create mechanisms that align individual incentives with collective goals, guiding participants toward cooperative behaviours and optimal allocations. Below, we describe the mechanism design in detail.

2.2 Mechanism Design: An Overview

Mechanism design is a fascinating discipline that explores the art of designing systems, rules, and protocols to achieve **desired outcomes in complex interactive settings**. It addresses the challenge of incentivizing self-interested individuals to behave in ways that align with broader objectives, such as efficiency, fairness, and optimal resource allocation. At its core, mechanism design aims to create rules and mechanisms that encourage participants to reveal their private information truthfully, cooperate with each other, and

make decisions that lead to desirable outcomes. It draws upon insights from economics, game theory, and social choice theory to design systems that navigate the complexities of individual incentives and collective goals.

Specifically, consider the resource allocation setting, where there are multiple players and multiple but finite items. Every player has a specific private valuation for these items. The social planner wants to find the optimal allocation. An allocation is considered optimal only when it satisfies certain fairness and/or efficiency criteria. There are two underlying problems; the first problem is of *preference elicitation* where the social planner must design a game to reveal the players' true valuation. In other words, at equilibrium, the players report their true valuations. The second problem is of *preference aggregation* or finding the optimal allocation given the player's valuations.

One of the fundamental concepts in mechanism design is *incentive compatibility*, which ensures that participants have a strong incentive to follow the desired behaviour outlined by the mechanism. By designing mechanisms that align participants' self-interest with the desired outcomes, mechanism designers can foster cooperation and achieve efficient allocations of resources. By combining theoretical analysis, mathematical modelling, and empirical studies, mechanism design provides a rigorous framework for understanding and shaping complex systems of human interaction. It enables us to create rules and mechanisms that incentivize desirable behaviours, mitigate conflicts of interest, and foster outcomes that benefit society as a whole.

Mechanism design is useful in various domains, including auctions, voting systems, market design, and public policy. It helps tackle challenges like designing fair and efficient auction mechanisms, developing voting systems that accurately represent societal preferences, or designing mechanisms for allocating limited resources to maximize social welfare. Below, we introduce preliminaries for mechanism design, which are followed by important definitions.

2.2.1 Preliminaries

Before delving into the intricacies of mechanism design, it is essential to establish a foundation of key concepts and principles. These preliminaries provide a framework for understanding the fundamental elements and goals of mechanism design.

Mechanism Design Environment: Below, we present a general setting to describe and analyze the mechanism design problems.

- **Set of Agents:** There are n agents, $N = \{1, 2, \dots, n\}$. The agents are rational and intelligent and capable of interacting strategically to make a collective decision.
- **Type:** Each agent has a private value or type that determines its preferences. It is denoted by θ_i , known as private value or type of agent i . The value of θ_i is known to agent i and may not be known to the other agents.
- **Type Set:** Θ_i denotes the set of private values of agent i , $\forall i \in N$. The type set is given by $\Theta = \Theta_1 \times \Theta_2 \times \dots \times \Theta_n$.
- **Set of Outcomes:** X is a set of outcomes or alternatives. The agents are supposed to make a collective choice from the set X .
- **Utility:** Agents have preferences over outcomes that are represented by a utility function $u_i : X \times \Theta_i \rightarrow \mathbb{R}$.
- **Strategy Set:** Each agent has a set of possible actions, called strategy and denoted by S_i . The collective strategy set is denoted by $S = S_1 \times S_2 \times \dots \times S_n$.
- **Common Prior:** We assume that there is a common prior $\phi = \phi_1 \times \phi_2 \times \dots \times \phi_n$.

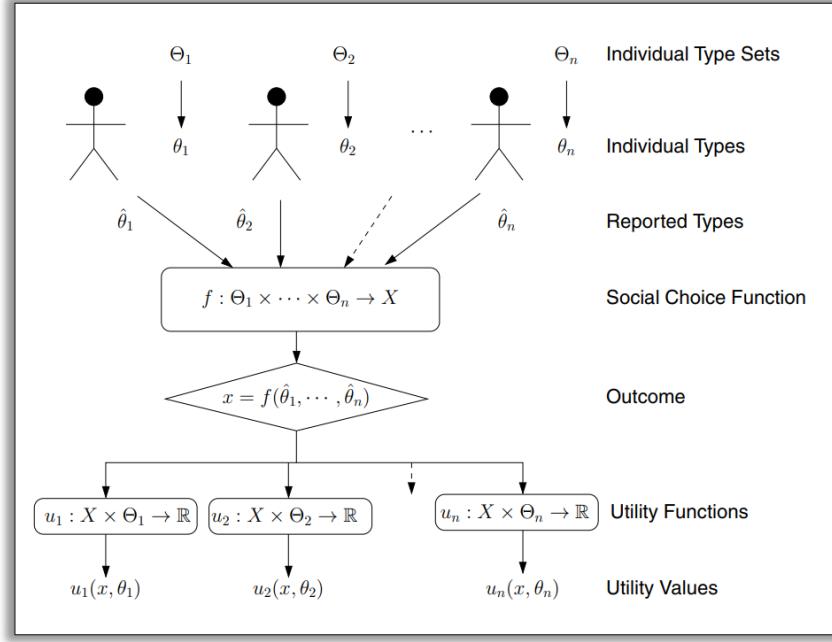


Figure 2.1: Mechanism Design Environment [Image Credit: © [2]]

2.2.2 Important Definitions in Mechanism Design

Mechanism design encompasses the study of properties that desirable mechanisms should possess to achieve desired outcomes. These properties serve as guidelines for designing effective mechanisms that incentivize desirable behaviours and achieve specific objectives. A pictorial representation of the mechanism design environment is presented in Figure 2.1. Below are some important aspects and definitions of mechanism design:

Definition 2.2.1: Social Choice Function (SCF) [15]

Suppose $N = \{1, 2, \dots, n\}$ is a set of agents with the type sets $\Theta_1, \Theta_2, \dots, \Theta_n$, respectively. Given a set of outcomes X , a ***social choice function*** is a mapping $f : \Theta_1 \times \Theta_2 \times \dots \times \Theta_n \rightarrow X$ that assigns to each possible type profile $\Theta_1 \times \Theta_2 \times \dots \times \Theta_n$,

an outcome from the set X . The outcome corresponding to a type profile is called a social choice or collective choice for that type profile.

In mechanism design, the social planner must address two key challenges to achieve the desired objectives. The first challenge involves extracting private information from each agent to make informed decisions. The second challenge is designing an appropriate mechanism to aggregate these reported types and produce an outcome that meets the specified criteria. These challenges, known as the preference elicitation problem and the preference aggregation problem, are detailed below.

Preference Elicitation Problem: Consider a social choice function $f : \Theta_1 \times \Theta_2 \times \dots \times \Theta_n \rightarrow X$. The types $\theta_1, \dots, \theta_n$ of individual agents are private information known only to them. For the social choice $f(\theta_1, \dots, \theta_n)$ to be made when the agents have types $\theta_1, \dots, \theta_n$, each agent needs to disclose its true type to the social planner. However, given a social choice function f , an agent might not find it beneficial to reveal this information truthfully. This challenge of making the agents reveal their true types is known as the preference elicitation problem or the information revelation problem.

Preference Aggregation Problem: Once all agents report their types, the profile of reported types must be converted into an outcome using the social choice function. Let θ_i be the true type of agent i , and $\hat{\theta}_i$ be the reported type of agent i (for $i = 1, \dots, n$). The process of determining $f(\hat{\theta}_1, \dots, \hat{\theta}_n)$ is referred to as the preference aggregation problem. The preference aggregation problem is often an optimization problem.

Thus, mechanism design can be seen as solving an optimization problem that is initially incomplete, requiring the specification to be elicited before solving the underlying decision problem. This specification elicitation is known as the preference or type elicitation problem. To gather truthful type information from agents, two main approaches are used: *direct mechanisms* and *indirect mechanisms*.

Definition 2.2.2: Direct Mechanism [15]

Suppose $f : \Theta_1 \times \Theta_2 \times \dots \times \Theta_n \rightarrow X$ is a social choice function. A **direct mechanism** (also called a direct revelation mechanism) corresponding to f consists of the tuple $(\Theta_1, \Theta_2, \dots, \Theta_n, f(.))$.

The definition of a direct mechanism implies that it directly asks the agents to reveal their true types to seek the type information of all agents.

Definition 2.2.3: Indirect Mechanism [15]

An **indirect mechanism** (also called an indirect revelation mechanism) consists of a tuple $(S_1, S_2, \dots, S_n, g(.))$ where S_i is a set of possible actions for agent $i, \forall i \in N$, and $g : S_1 \times S_2 \times \dots \times S_n \rightarrow X$ is a function that maps each action profile to an outcome.

An indirect mechanism provides a choice of actions to each agent (strategy) and specifies an outcome for each action profile.

A mechanism is a framework or system governed by a set of rules that define the actions available to the agents and outline how these action profiles are converted into outcomes. Given a set of available actions, an agent can decide what action she wants to play at a given instance, which is governed by the type of the agent.

Definition 2.2.4: Strategy in Bayesian Game [15]

For $i = 1, 2, \dots, n$, a **strategy** s_i for agent i in the induced **Bayesian game** is a mapping $s_i : \Theta_i \rightarrow S_i$. Thus, given a type $\theta_i \in \Theta_i$, $s_i(\theta_i)$ will give the action of player i corresponding to θ_i . The strategy $s_i(.)$ will specify actions corresponding to types.

In the auction scenario, the bid b_i of player i is a function of the valuation θ_i . For example, $b_i(\theta_i) = \alpha_i \theta_i$ is a particular strategy for player i (also known as **scale-based bidding strategy**).

The **preference elicitation** problem focuses on gathering truthful information from agents regarding their private types. To elicit truthful information from agents, the goal is to make truth-telling their best strategy, aligning with the assumptions of rationality and intelligence. One way to achieve this is by providing appropriate incentives, which is the essence of incentive compatibility. Incentive compatibility refers to designing mechanisms that motivate agents to disclose their true types. Broadly, there are two types of incentive compatibility: (1) **Dominant Strategy Incentive Compatibility (DSIC)**, where truth-telling is the best response for each agent regardless of what others report; and (2) **Bayesian Nash Incentive Compatibility (BIC)**, where truth-telling is the best response based on the agent's expectations of the other agents' types. Since the concept of truth-telling pertains to agents' types, only direct revelation mechanisms are considered when formalizing incentive compatibility.

Definition 2.2.5: Incentive Compatibility (IC) or Strategy-proof or Truthful [15]

A social choice function $f : \Theta_1 \times \Theta_2 \times \dots \times \Theta_n \rightarrow X$ is said to be **incentive compatible** (or truthfully implementable) if the Bayesian game induced by the direct mechanism $\mathbb{D} = ((\Theta_i)_{i \in N}, f())$ has a pure strategy equilibrium $s^* = (s_1^*, s_2^*, \dots, s_n^*)$ in which $s_i^*(\theta_i) = \theta_i, \forall \theta_i \in \Theta_i, \forall i \in N$.

Definition 2.2.6: Dominant Strategy Incentive Compatibility (DSIC) [15]

A social choice function $f : \Theta_1 \times \Theta_2 \times \dots \times \Theta_n \rightarrow X$ is said to be **dominant strategy incentive compatible** (or truthfully implementable in dominant strategies) if the direct revelation mechanism $\mathbb{D} = ((\Theta_i), i \in N, f(.))$ has a weakly dominant strategy equilibrium $s^*(.) = (s_1^*(.), \dots, s_n^*(.))$ in which $s_i^*(\theta_i) = \theta_i, \forall \theta_i \in \Theta_i, \forall i \in N$.

That is, truth revelation by each agent constitutes a weakly dominant strategy equilibrium of the game induced by \mathbb{D} .

Definition 2.2.7: Bayesian Incentive Compatibility (BIC) [15]

A social choice function $f : \Theta_1 \times \Theta_2 \times \dots \times \Theta_n \rightarrow X$ is said to be ***Bayesian incentive compatible*** (or truthfully implementable in Bayesian Nash equilibrium) if the direct revelation mechanism $\mathbb{D} = ((\Theta_i), i \in N, f(\cdot))$ has a Bayesian Nash equilibrium $s^*(\cdot) = (s_1^*(\cdot), \dots, s_n^*(\cdot))$ in which $s_i^*(\theta_i) = \theta_i, \forall \theta_i \in \Theta_i, \forall i \in N$.

That is, truth revelation by each agent constitutes a weakly Bayesian Nash equilibrium of the game induced by \mathbb{D} .

Properties of a Social Choice Function

In this section, we present some of the important properties a social planner would like the social choice function to possess.

Definition 2.2.8: Pareto Optimality (PO) [15]

For a social choice function $f : \Theta \rightarrow X$, the outcome $f(\theta)$ is ***Pareto optimal*** if there does not exist any $x \in X$ such that:

$$u_i(x, \theta_i) \geq u_i(f(\theta), \theta_i), \forall i \in N \text{ and } u_i(x, \theta_i) > u_i(f(\theta), \theta_i), \text{ for some } i \in N$$

That is, there is no other outcome where any agent can be made better off without making some other agent worse off.

Definition 2.2.9: Ex-Post Efficiency (EPE) [15]

A social choice function $f : \Theta \rightarrow X$ is said to be ***ex-post efficient*** (or Paretian) if for every profile of agents' types, $\theta \in \Theta$, the outcome $f(\theta)$ is a Pareto optimal outcome.

That is, if an SCF is ex-post efficient, there does not exist any outcome that is better for all the players.

Definition 2.2.10: Dictatorship [15]

A social choice function $f : \Theta \rightarrow X$ is said to be **dictatorial** if there exists an agent $d \in N$ such that $\forall \theta \in \Theta$,

$$u_d(f(\theta), \theta_d) \geq u_d(x, \theta_d) \quad \forall x \in X$$

That is, a dictator is an agent whose preferences always determine the outcome of the social choice function, ensuring that the chosen outcome is always the one most preferred by that agent.

In mechanism design, dominant strategy incentive compatibility (DSIC) is a highly desirable property for social choice functions. However, due to its strict nature, achieving DSIC may prevent other desirable properties from being fulfilled. Furthermore, below we introduce the Gibbard-Satterthwaite impossibility theorem (**GS theorem**), which demonstrates that in an unrestricted utility environment, enforcing DSIC will inevitably lead to a social choice function that is dictatorial.

Theorem 2.1. The Gibbard-Satterthwaite Impossibility Theorem [15]

Consider a social choice function $f : \Theta \Rightarrow X$. Suppose that

- The outcome set X is finite and $X \geq 3$,
- $\mathcal{R}_i = \mathcal{P} \quad \forall i \in N$,
- f is an onto mapping; that is, the image of SCF $f(\cdot)$ is the set X .

Then the social choice function f is dominant strategy incentive compatible iff it is dictatorial.

Thus, to overcome the impossibility imposed by the GS theorem, we apply further restrictions on the utility environment and introduce the *quasilinear environment*. In this environment, agents' preferences are assumed to be quasilinear, meaning their utility functions can be expressed as the difference between the value they assign to an outcome and the payment they make. This allows for mechanisms where monetary transfers can be used as incentives for truthful reporting. Unlike the general setting where any non-dictatorial social choice function is susceptible to strategic manipulation, the quasilinear environment supports mechanisms, such as the Vickrey-Clarke-Groves (VCG) mechanism, that achieve both incentive compatibility and efficient outcomes. By integrating allocation and payment rules, these mechanisms ensure that agents maximize their utility by reporting their true preferences, thus sidestepping the impossibility result of the GS theorem.

The Quasi-linear Environment: In the quasi-linear environment, an outcome $x \in X$ is represented as a vector of the form $x = (k, t_1, t_2, \dots, t_n)$, where, $k \in K$. K is a set of allocations or project choices, which is assumed to be finite. The term $t_i \in \mathbb{R}$ denotes the payments or monetary transfer to agent i (If $t_i > 0$, agent i receives a payment t_i and vice versa). Thus, an outcome is decomposed into two parts, (i) allocation and (ii) payment. The set of alternatives X is therefore,

$$X = \{(k, t_1, t_2, \dots, t_n) : k \in K; t_i \in \mathbb{R}; \forall i \in N\}$$

Below, we present a few other important properties a social planner would like the social choice function to possess within the quasi-linear environment.

Definition 2.2.11: Week Budget Balance (WBB) [15]

Generally, in the quasi-linear environment, it is assumed that the n agents have no external source of funding; thus, we make an assumption that $\sum_{i \in N} t_i \leq 0$. This condition is known as the *weak budget balance* condition.

Thus, with the weak budget balance condition, the set of alternatives can be represented as: $X = \{(k, t_1, t_2, \dots, t_n) : k \in K; t_i \in \mathbb{R}; \forall i \in N; \sum_{i \in N} t_i \leq 0\}$

Definition 2.2.12: Strictly Budget Balance (SBB) [15]

In the Definition 2.2.2, if we make an assumption that $\sum_{i \in N} t_i = 0$. This condition is known as the *strictly budget balance* condition.

The strictly budget balance condition implies that the in-flow and the out-flow of the money are the same in the system; thus, the net payments are zero.

Definition 2.2.13: Individual Rationality (IR) [15]

Individual rationality of a social choice function essentially means that each agent gains a utility that is no less than he would get without participating in a mechanism that implements the social choice function. That is,

$$v_i(k(\theta), \theta_i) - t_i \geq 0, \forall k \in K, \forall i \in N$$

Definition 2.2.14: Allocative Efficiency (AE) [15]

We say that a social choice function $f() = (k(.), t_1(.), t_2(.), \dots, t_n(.))$ is *allocatively efficient* if for each $\theta \in \Theta$, $k(\theta)$ satisfies the following condition,

$$k(\theta) \in \operatorname{argmax}_{k \in K} \sum_{i=1}^n v_i(k, \theta_i)$$

Equivalently,

$$\sum_{i=1}^n v_i(k(\theta), \theta_i) = \max_{k \in K} \sum_{i=1}^n v_i(k, \theta_i)$$

In other words, the above definition says that for every $\theta \in \Theta$, the allocation $k(\theta)$ will maximize the sum of the values of the players, meaning every allocation is a value-

maximizing allocation. If the allocations are AE, then the objects are allocated to the players who value the objects most.

However, the **Myerson-Satterthwaite Theorem** is an impossibility result that says, in a bilateral trade setting, there is no SCF that satisfies AE, SBB, BIC, and Interim IR simultaneously. Below is a formal statement,

Theorem 2.2. The Myerson-Satterthwaite Impossibility Theorem [15]

Consider a bilateral trade setting in which the buyer and seller are risk-neutral, the valuations θ_1 and θ_2 are drawn independently from the intervals $[l_b, h_b] \in \mathbb{R}$ and $[l_s, h_s] \in \mathbb{R}$ with strict positive densities, and $[l_b, h_b] \cap [l_s, h_s] \neq \emptyset$. Then, there is no Bayesian incentive compatible (BIC) social choice function that is ex-post efficient (AE) and gives every buyer and every seller nonnegative expected utilities (IR) from participation.

2.2.3 Auctions

An auction is a mechanism to allocate a set of goods to a set of bidders on the basis of bids announced by the bidders [16]. In today's world, auctions are ubiquitous, be it in stock market trading, online advertisements, spectrum auctions, and many more. Auctions can be of varied types depending on the use case. Each auction has its own set of rules and properties. Below, we discussed some of the most important types of auctions.

2.2.3.1 Auction Types and Desirable Properties

We can classify auctions based on six major criteria as suggested by Kalagnanam and Parkes's framework [17].

1. **Resource:** Auctions can be classified based on the number of items for sale, which could be single or multiple items, with single or multiple units of each item.

2. **Market Structure:** Based on the market structure, auctions can be classified into three categories, (i) A forward auction consists of one seller and multiple buyers who submit their bids; (ii) A reverse auction has one buyer and multiple sellers submitting their asks, and (iii) A double auction comprising of multiple buyers and sellers placing their bids and asks.
3. **Preference Structure:** The preference defines an agent's utility for various outcomes. For example, in the case of multiple units of an item, there can be a decreased marginal utility for additional items.
4. **Bid Structure:** The bid structure within the auction provides the flexibility to agents to express their resource requirements. In some auction mechanisms, agents only need to submit the price, whereas, in some other mechanisms, both price and quantity need to be submitted in the bid.
5. **Winner Determination:** The auctions can be categorized based on the auction clearing rules and allocation rules. For example, (i) In a forward auction, the winner determination rule assigns the items to an optimal set of buyers, (ii) In a reverse auction, it selects an optimal set of sellers to be contracted for supplying required items, and (iii) In a double auction, the winner determination finds an optimal matching between buyers and sellers.
6. **Information Feedback:** An auction mechanism may contain information feedback or may not have any information feedback. Feedback can be a price signal or a tentative allocation, based on which agents can plan for future auctions.

Any auction mechanism must fulfil some of the desirable properties discussed in Section 2.2.2, namely, Equilibrium, Incentive Compatibility, Allocative Efficiency, Individual Rationality, Budget Balance, Revenue Maximization, and Fairness. **Revenue maximization** refers to the objectives of buyers and sellers; the buyers would aim to procure items

at a minimum cost, while sellers would want to maximize their total revenue. **Fairness** ensures that the winner determination algorithms, especially those based on heuristics, produce consistent results in which all deserving bidders should get a fair chance to be victorious. **Note that it is not possible to have an auction mechanism satisfying all the desirable properties (Theorem 2.2)**. Thus, an auction designer needs to carefully select a maximal subset of these properties that can be simultaneously achieved. Below, we explain some of the well-known auction mechanisms in detail. For more information about various types of auctions, there are some great review articles [18, 19, 20].

2.2.3.2 Single Sided Auctions

In **single-sided auctions**, only one side of agents (either bidders or sellers) deviate from their true valuations while placing bids. For example, there could be a mechanism where the seller reveals its true valuation for the item/items as the ask price, and buyers strategize to place bids to procure the item/items, and vice versa. These auctions can be further classified into **second-price** [21] and **first-price** [22] auctions based on their winner-determination rules. Below, we explain both the mechanism, as well as the corresponding strategies for the agents.

Second Price Auctions

Consider the auction setting, where there is 1 seller and 2 buyer; both buyers are competing for a single item. Thus, the set of agents can be denoted as $N = \{0, 1, 2\}$. 0 represents the seller, whereas, 1 and 2 represent two buyers. The types of each player are denoted by θ_0 , θ_1 , and θ_2 , respectively; thus, the type set becomes $\Theta = \{\theta_0, \theta_1, \theta_2\}$. The set of outcomes X is given below,

$$X = \{(k_0, k_1, k_2, t_0, t_1, t_2) : k_i \in \{0, 1\}; k_0 + k_1 + k_2 = 1; t_i \in \mathbb{R}; \forall i \in N\}$$

As this is a quasi-linear setting, the utility function for both the buyers is given by,

$$u_i((k_0, k_1, k_2, t_0, t_1, t_2), \theta_i) = k_i \times \theta_i + t_i, \quad i = 1, 2$$

The utilities depend upon the strategies of the buyers. As the seller is non-strategic, $S_0 = \theta_0$; while buyers can bid any positive value in the auction, thus, $S_1 \in \mathbb{R}_+$ and $S_2 \in \mathbb{R}_+$. Since S_0 is a singleton, the ask of the seller is common knowledge. Based on this description, the following are the winner-determination rules. Let's assume the allocation rule for the bid profiles $b = (b_0, b_1, b_2) \in S_0 \times S_1 \times S_2$ is given by,

$$g(b) = (k_0(b), k_1(b), k_2(b), t_0(b), t_1(b), t_2(b)), \text{ such that } \forall (b_0, b_1, b_2) \in S_0 \times S_1 \times S_2,$$

$$k_0(b_0, b_1, b_2) = 0$$

$$\begin{aligned} k_1(b_0, b_1, b_2) &= 1, \text{ if } b_1 \geq b_2 \\ &= 0, \text{ otherwise} \end{aligned}$$

$$\begin{aligned} k_2(b_0, b_1, b_2) &= 1, \text{ if } b_2 > b_1 \\ &= 0, \text{ otherwise} \end{aligned}$$

$$t_1(b_0, b_1, b_2) = -k_1(b_0, b_1, b_2)b_2$$

$$t_2(b_0, b_1, b_2) = -k_2(b_0, b_1, b_2)b_1$$

$$t_0(b_0, b_1, b_2) = -(t_1(b_0, b_1, b_2) + t_2(b_0, b_1, b_2))$$

Theorem 2.3. *For a single-seller two-buyer single-unit second-price auction where the seller is non-strategic and has type θ_S ; $\theta_1 \sim \mathbf{U}[0, 1]$ and $\theta_2 \sim \mathbf{U}[0, 1]$, respectively; when the buyers deploy scale based bidding strategies b_1 and b_2 , then the second-price auction is truthful at the equilibrium; thus, both buyers bid their true types at the equilibrium ($b_1 = \theta_1$ and $b_2 = \theta_2$) [15]^a.*

^aThe proof is taken from the book “Game Theory and Mechanism Design” by Prof. Narahari [15]

Proof. As stated in the theorem, buyer i has valuation θ_i ; assume that buyer 1 places the bid b_1 . Now, let the maximum bid of the buyer 2 is t . The bid t is independent of buyer 1's bid. Now, there are two cases, (i) $\theta_1 \leq t$, and (ii) $\theta_1 > t$. Let's analyze each case individually.

Proof. Case 1: $\theta_1 \leq t$

In this case, buyer 1's valuation is less than the maximum bid of buyer 2. Now, it can choose to play truthfully and bid its true valuation θ_1 . As $\theta_1 \leq t$, buyer 2 wins the item, and the utility of buyer 1 remains zero. However, if buyer 1 decides to play strategically and plays a bid higher than t , then the utility of buyer 1 becomes $u_1 = \theta_1 - t$. But, as $\theta_1 \leq t$, u_1 is negative.

Case 2: $\theta_1 > t$

In this case, buyer 1's valuation is more than the maximum bid of buyer 2. Now, it can choose to play truthfully and bid its true valuation θ_1 . As $\theta_1 > t$, buyer 1 wins the item, and the utility of buyer 1 becomes $u_1 = \theta_1 - t$. As $\theta_1 > t$, u_1 is positive. Now, suppose buyer 1 decides to play strategically and plays a bid lower than θ_1 but higher than t . In this case, the utility u_1 does not change. However, if the bid becomes lower than t , then buyer 2 wins the item, and the utility of buyer 1 reduces to zero.

The same analysis can be carried out for buyer 2 as well. Thus, bidding truthfully yields the highest utility for both buyers and by not playing truthfully, a buyer can not increase its utility in any case. $b_1 = \theta_1$ and $b_2 = \theta_2$ form equilibrium strategies for buyer 1 and buyer 2, respectively. □

First Price Auctions

Consider an auction setting similar to the second-price auction, where there is 1 seller and 2 buyers; both buyers are competing for a single item in the first-price auction. Thus,

the set of agents can be denoted as $N = \{0, 1, 2\}$. 0 represents the seller, whereas, 1 and 2 represent two buyers. The types of each player are denoted by θ_0 , θ_1 , and θ_2 , respectively; thus, the type set becomes $\Theta = \{\theta_0, \theta_1, \theta_2\}$. The set of outcomes X is given below,

$$X = \{(k_0, k_1, k_2, t_0, t_1, t_2) : k_i \in \{0, 1\}; k_0 + k_1 + k_2 = 1; t_i \in \mathbb{R}; \forall i \in N\}$$

As discussed, k_i denotes the allocation of the item to each agent; as there is only a single item for trading, k_i can be either 0 (item is not allocated to the agent) or 1 (item is allocated to the agent); thus, $k_i \in \{0, 1\}$. t_i denotes the payments to agents; $t_0 \geq 0$ as the seller receives money if the trade happens, whereas $t_1, t_2 \leq 0$ as the buyers pay money when the trade happens. As this is a quasi-linear setting, the utility function for both the buyers is given by,

$$u_i((k_0, k_1, k_2, t_0, t_1, t_2), \theta_i) = k_i \times \theta_i + t_i, \quad i = 1, 2$$

The utilities depend upon the strategies of the buyers. As the seller is non-strategic, $S_0 = \theta_0$; while buyers can bid any positive value in the auction, thus, $S_1 \in \mathbb{R}_+$ and $S_2 \in \mathbb{R}_+$. Since S_0 is a singleton, the ask of the seller is common knowledge. Based on this description, the following are the winner-determination rules. Let's assume the allocation rule for the bid profiles $b = (b_0, b_1, b_2) \in S_0 \times S_1 \times S_2$ is given by,

$$g(b) = (k_0(b), k_1(b), k_2(b), t_0(b), t_1(b), t_2(b)), \text{ such that } \forall (b_0, b_1, b_2) \in S_0 \times S_1 \times S_2,$$

$$\begin{aligned} k_0(b_0, b_1, b_2) &= 0 \\ k_1(b_0, b_1, b_2) &= 1, \text{ if } b_1 \geq b_2 \\ &\quad = 0, \text{ otherwise} \\ k_2(b_0, b_1, b_2) &= 1, \text{ if } b_2 > b_1 \\ &\quad = 0, \text{ otherwise} \end{aligned}$$

$$\begin{aligned}
t_1(b_0, b_1, b_2) &= -k_1(b_0, b_1, b_2)b_1 \\
t_2(b_0, b_1, b_2) &= -k_2(b_0, b_1, b_2)b_2 \\
t_0(b_0, b_1, b_2) &= -(t_1(b_0, b_1, b_2) + t_2(b_0, b_1, b_2))
\end{aligned}$$

Theorem 2.4. For a single-seller two-buyer single-unit first-price auction where the seller is non-strategic and has type θ_S ; $\theta_1 \sim U[0, 1]$ and $\theta_2 \sim U[0, 1]$, respectively; when the buyers deploy scale based bidding strategies b_1 and b_2 , then the first-price auction is not truthful at the equilibrium. The equilibrium strategies for the buyers are given by $b_1 = \theta_1/2$ and $b_2 = \theta_2/2$ [15]^a.

^aThe proof is taken from the book “Game Theory and Mechanism Design” by Prof. Narahari [15]

Proof. Based on the information given in the theorem, let the bid of the 2nd buyer is $b_2 = \alpha_2\theta_2$, where $\alpha_2 \in (0, 1]$. As $\theta_2 \sim U[0, 1]$, the maximum bid of buyer 2 is α_2 . Buyer 1 knows this; thus, buyer 1 bid is capped at α_2 , so, $b_1 \in [0, \alpha_2]$.

Now, as discussed above, buyer 1 gets the item if $b_1 \geq b_2$; thus, $b_1 \geq \alpha_2\theta_2$, which implies, $\theta_2 \leq \frac{b_1}{\alpha_2}$. Now, the expected utility of buyer 1 is,

$$\begin{aligned}
U_1(b_1; \theta_1) &= \mathbb{E}_{b_2} u_1(b_1, b_2; \theta_1) \\
&= \mathbb{E}_{\theta_2} u_1(b_1, \alpha_2\theta_2; \theta_1) \\
&= (\theta_1 - b_1) Pr(b_1 \geq b_2) \\
&= (\theta_1 - b_1) Pr(\theta_2 \leq \frac{b_1}{\alpha_2}) \\
&= (\theta_1 - b_1) \frac{b_1}{\alpha_2}
\end{aligned}$$

Proof. Now, to find the bid b_1 that maximizes the expected utility of buyer 1,

$$\frac{dU}{db_1}(\theta_1 - b_1) \frac{b_1}{\alpha_2} = 0 \Rightarrow b_1 = \frac{\theta_1}{2}$$

So, we can write,

$$\begin{aligned} b_1 &= \theta_1/2, \text{ if } \theta_1/2 \leq \alpha_2 \\ &= \alpha_2, \text{ otherwise} \end{aligned}$$

$$\begin{aligned} b_2 &= \theta_2/2, \text{ if } \theta_2/2 \leq \alpha_1 \\ &= \alpha_1, \text{ otherwise} \end{aligned}$$

The above equations imply $\alpha_1 = \alpha_2 = 1/2$. As $\theta_1, \theta_2 \sim \mathbf{U}[0, 1]$, $b_1 = \theta_1/2$ and $b_2 = \theta_2/2$ form equilibrium strategies for buyer 1 and buyer 2, respectively. \square

2.2.3.3 Double Sided Auctions

In **double-sided auctions**, both buyers and sellers deviate from their true valuations while placing bids in order to maximize their gain from the auction. Similar to single-sided auctions, these auctions, too, can be further classified into second-price and first-price auctions based on their winner determination rules. We particularly discuss two popular mechanisms of each type, the **VCG auction** [21, 23, 24], which is a second-price auction, and the **k-double auction** [25], which is a first-price auction. Below, we explain both the mechanism, as well as the corresponding strategies for the agents.

VCG Auctions

Consider an auction setting where there is 1 seller and 1 buyer trading for a single item in the VCG auction. Thus, the set of agents can be denoted as $N = \{S, B\}$. S represents the seller, and B represents the buyer. The types of each player are denoted by θ_S and θ_B , respectively; thus, the type set becomes $\Theta = \{\theta_S, \theta_B\}$. The set of outcomes X is given

below,

$$X = \{(k_S, k_B, t_S, t_B) : k_i \in \{0, 1\}; k_S + k_B = 1; t_i \in \mathbb{R}; \forall i \in N\}$$

The strategies of the seller and the buyer are given by $S_S = \mathbb{R}_+$ and $S_B = \mathbb{R}_+$. Based on this description, the following are the winner-determination rules. Let's assume the allocation rule for the bid profiles $b = (b_S, b_B) \in S_S \times S_B$ is given by,

$$g(b) = (k_S(b), k_B(b), t_S(b), t_B(b)), \text{ such that } \forall (b_S, b_B) \in S_S \times S_B$$

$$k_S(b_S, b_B) = 1, \text{ if } b_S > b_B$$

$$= 0, \text{ otherwise}$$

$$k_B(b_S, b_B) = 1, \text{ if } b_B \geq b_S$$

$$= 0, \text{ otherwise}$$

$$t_S(b_S, b_B) = k_B(b_S, b_B)b_B$$

$$t_B(b_S, b_B) = -k_B(b_S, b_B)b_S$$

The utility functions for the buyer and the seller are given by,

$$u_B((k_S, k_B, t_S, t_B), \theta_B) = k_B\theta_B + t_B$$

$$u_S((k_S, k_B, t_S, t_B), \theta_S) = -k_B\theta_B + t_S$$

The utility of the seller has a negative sign as the seller has a negative valuation for the item (because the seller wants to sell it).

Theorem 2.5. *For a single-seller single-buyer single-unit VCG auction that has types θ_S and θ_B , respectively, the VCG payments to the seller and buyer are given by t_S and t_B , respectively. These payments are $t_S = \theta_B$ and $t_B = -\theta_S$. The VCG auction mechanism is not a balanced budget.*

Proof. As per the VCG payment rule,

$$\text{VCG payment formula} = \sum_{j \neq i} V_j(k^*(\theta), \theta_j) - \sum_{j \neq i} V_j(k_{-i}^*(\theta_{-i}), \theta_j)$$

As per the VCG payment rule,

$$\text{the payment to the seller} = \theta_B$$

$$\text{the payment to the buyer} = -\theta_S$$

Hence, proved.

However, the VCG mechanism is not budget balanced. Here,

$$t_S(b_S, b_B) + t_B(b_S, b_B) = k_B(b_S, b_B)(b_B - b_S) = T$$

So, if $b_B < b_S$, then $k_B(b_S, b_B) = 0$,

otherwise, $T = b_B - b_S \geq 0$

As $T \geq 0$, the auctioneer has to fund the auction. Thus, the VCG auction is not budget-balanced. \square

k-Double Auctions

Consider an auction setting where there is 1 seller and 1 buyer trading for a single item in the k -double auction. Thus, the set of agents can be denoted as $N = \{S, B\}$. S represents the seller, and B represents the buyer. The types of each player are denoted by θ_S and θ_B , respectively; thus, the type set becomes $\Theta = \{\theta_S, \theta_B\}$. The set of outcomes X is given below,

$$X = \{(k_S, k_B, t_S, t_B) : k_i \in \{0, 1\}; k_S + k_B = 1; t_i \in \mathbb{R}; \forall i \in N\}$$

Let's assume both buyer and seller deploy a scale-based strategy. The strategies of the seller and the buyer are given by $S_S = \mathbb{R}_+$ and $S_B = \mathbb{R}_+$. The bids of the buyer and seller are $b_S = \alpha_S \theta_S$ and $b_B = \alpha_B \theta_B$, respectively. Based on this description, the following

are the winner-determination rules. Let's assume the allocation rule for the bid profiles $b = (b_S, b_B) \in S_S \times S_B$ is given by,

$$g(b) = (k_S(b), k_B(b), t_S(b), t_B(b)), \text{ such that } \forall (b_S, b_B) \in S_S \times S_B$$

$$k_S(b_S, b_B) = 1, \text{ if } b_S > b_B$$

$$= 0, \text{ otherwise}$$

$$k_B(b_S, b_B) = 1, \text{ if } b_B \geq b_S$$

$$= 0, \text{ otherwise}$$

$$t_S(b_S, b_B) = k_B(b_S, b_B)(k * b_B + (1 - k) * b_S)$$

$$t_B(b_S, b_B) = -k_B(b_S, b_B)(k * b_B + (1 - k) * b_S)$$

The utility functions for the buyer and the seller are given by,

$$u_B((k_S, k_B, t_S, t_B), \theta_B) = k_B \theta_B + t_B$$

$$u_S((k_S, k_B, t_S, t_B), \theta_S) = -k_B \theta_B + t_S$$

Theorem 2.6 (*Susobhan et. al. [26]*). *For a single-buyer, single-seller, single-unit k -double auction, where $k = 0.5$, and buyer's and seller's true types are drawn from a $0 - 1$ uniform distribution, $\theta_S \sim \mathbf{U}[0, 1]$ and $\theta_B \sim \mathbf{U}[0, 1]$, if they deploy scaling-based bidding strategies b_B and b_S and fix their scaling factors α_B and α_S before seeing their true types, then $\alpha_S = 1$ and $\alpha_B = \frac{2}{3}$ constitute a BNE.*

Proof. We make further assumptions in Equation 2.1, which says that the buyer's bid (seller's ask) at any point will be less (higher) than or equal to the highest (lowest) possible seller's ask (buyer's bid).

$$\frac{\alpha_B}{\alpha_S} \theta_B \leq 1; \quad \frac{\alpha_S}{\alpha_B} \theta_S \geq 0 \quad (2.1)$$

Proof. Thus, the utility of the buyer, if its bid gets cleared, is denoted by the difference between true valuation and clearing price. Given the true types are picked over a distribution, the expected utility is computed as:

$$u_B = \int_0^{\frac{\alpha_B}{\alpha_S} \theta_B} \left(\theta_B - \left(\frac{\alpha_B \theta_B + \alpha_S \theta_S}{2} \right) \right) d\theta_S$$

$$u_B = \left(1 - \frac{\alpha_B}{2} \right) \frac{\alpha_B}{\alpha_S} \theta_B^2 - \frac{1}{4} \frac{\alpha_B^2 \theta_B^2}{\alpha_S}$$

Now, assuming that the buyer decides to fix its α_B before even seeing its own type, then its utility is given by:

$$U_B = \int_0^1 u_B d\theta_B$$

$$U_B = \left(1 - \frac{\alpha_B}{2} \right) \frac{\alpha_B}{\alpha_S} \frac{1}{3} - \frac{1}{12} \frac{\alpha_B^2}{\alpha_S}$$

Now, differentiating with respect to α_B and equating to 0 to find maxima:

$$\frac{\partial U_B}{\partial \alpha_B} = 0 \Rightarrow \frac{1}{3\alpha_S} - \frac{\alpha_B}{3\alpha_S} - \frac{\alpha_B}{6\alpha_S} = 0 \Rightarrow \alpha_B = \frac{2}{3}$$

Performing a similar analysis for the seller results in,

$$u_S = \int_{\frac{\alpha_S}{\alpha_B} \theta_S}^1 \left(\left(\frac{\alpha_B \theta_B + \alpha_S \theta_S}{2} \right) - \theta_S \right) d\theta_B$$

$$u_S = \frac{\alpha_B}{4} \left(1 - \left(\frac{\alpha_S}{\alpha_B} \theta_S \right)^2 \right) + \left(\frac{\alpha_S}{2} - 1 \right) \left(1 - \frac{\alpha_S}{\alpha_B} \theta_S \right) \theta_S$$

Now, assuming that the seller decides to fix its α_S before even seeing its own type, then its utility is given by U_S . Then, differentiating it with respect to α_B and equating it to 0 to find maxima.

$$U_S = \int_0^1 u_S d\theta_S$$

$$U_S = \frac{\alpha_B}{4} - \frac{\alpha_S^2}{12\alpha_B} + \left(\frac{\alpha_S}{2} - 1 \right) \left(\frac{1}{2} - \frac{\alpha_S}{3\alpha_B} \right)$$

Proof.

$$\frac{\partial U_S}{\partial \alpha_S} = 0 \Rightarrow \alpha_S = 1$$

Hence, proved. □

However, the aforementioned theorem focuses solely on the single-unit k -double auction. In contrast, we present a generalized version of the theorem that applies to the n -unit k -double auction, as discussed in Chapter 5. As the number of players or units increases, the analysis of the k -double auction becomes increasingly intractable. In such cases, **learning-based methods**—specifically **reinforcement learning (RL) algorithms**—can be highly effective. In Chapter 5, we demonstrate how scaling factors can be determined in complex, real-world scenarios involving multi-player, multi-unit k -double auctions.

2.3 Summary

This chapter provides an overview of the game theory and mechanism design concepts, which will be useful in the later chapters. More specifically, we presented details about the use cases of game theory, representation of a game, and equilibrium concepts. In the mechanisms design framework, we discussed the framework in detail, including all the notations and desirable properties of a mechanism. Finally, we provided an overview of various auction mechanisms and a detailed discussion of some of the popular auction mechanisms and the equilibrium strategies for agents in those auctions. However, deriving analytical solutions becomes increasingly challenging as the complexity of the problem grows. In such cases, RL offers a powerful alternative for addressing these limitations. In the next chapter, we provide a comprehensive exploration of RL techniques and their application to overcome these challenges.

Chapter 3

Reinforcement Learning

“...What we want is a machine that can learn from EXPERIENCE”

– Alan Turing, 1947

Reinforcement Learning achieves just this.

Reinforcement Learning (RL) is a rapidly advancing area of machine learning focused on training intelligent agents to make sequential decisions by interacting with an environment. Unlike supervised learning, where agents learn from labeled data, RL agents learn through trial and error, guided by rewards or penalties. This chapter provides a comprehensive overview of RL, highlighting its theoretical foundations, algorithms, and applications across diverse domains. Key concepts such as **Markov Decision Processes (MDPs)**, **policy optimization**, **value functions**, and **exploration-exploitation trade-offs** are introduced to lay a solid foundation for understanding RL frameworks.

The chapter explores classical RL approaches like Q-learning and SARSA, alongside advanced methods, including deep reinforcement learning techniques such as Actor-Critic architectures. Emphasis is placed on the role of function approximation using neural networks to handle large state and action spaces, enabling RL applications in complex, high-dimensional environments. We also introduce Monte-Carlo Tree Search (MCTS), which combines the precision of tree-search with random sampling. Finally, we provide an overview of **Multi-armed Bandits (MABs)** and its derivatives. We introduce several important bandits algorithms that we use in our research work.

3.1 Reinforcement Learning: An Overview

Reinforcement learning is a branch of machine learning that focuses on developing intelligent agents capable of making *sequential decisions* and *learning from interactions* with an *environment*. Inspired by the principles of behavioral psychology, reinforcement learning is based on the concept of an *agent* receiving feedback in the form of rewards or punishments based on its actions. By iteratively interacting with the environment and maximizing cumulative rewards, the agent learns to make optimal decisions in order to achieve *long-term goals*. Below, we present some of the example scenarios in which RL has made an enormous impact. We take motivation from the below success stories to utilize RL in our problem domain.

- **Finance and Trading:** Reinforcement learning is used to develop trading strategies in financial markets. Agents learn to buy/sell decisions based on market data, historical trends, and reward signals tied to profit or risk objectives [26, 27, 28, 29, 30, 31].
- **Game Playing:** Reinforcement learning has achieved remarkable success in game playing. For instance, AlphaGo, developed by DeepMind, used reinforcement learning

to defeat world champion Go players. Similarly, AlphaZero mastered chess and shogi by learning solely through self-play and reinforcement learning techniques [32, 33, 34].

- **Autonomous Vehicles:** Reinforcement learning plays a role in developing self-driving cars. Agents learn to make decisions such as lane changing, acceleration, and braking based on sensor inputs. They learn through simulations and real-world experiences to improve safety, efficiency, and decision-making in diverse driving situations [35, 36].
- **Robotics:** Reinforcement learning is employed in training robots to perform complex tasks. Robots can learn to grasp objects, navigate through environments, or even manipulate objects with dexterity. This enables them to adapt to real-world scenarios and learn from experience to improve their performance [37, 38].
- **Control Systems:** Reinforcement learning is applied in controlling complex systems, such as power grids or chemical plants. Agents learn optimal control policies to manage and optimize system operations while considering constraints and achieving predefined objectives [39, 40].
- **Healthcare:** Reinforcement learning has been explored for personalized treatment recommendation and medical decision-making. Agents learn to make treatment decisions based on patient data and clinical outcomes, taking into account individual characteristics and optimizing patient health [41].

The core idea behind reinforcement learning is to create an agent that can learn from trial and error, exploring different actions and observing their consequences. The agent takes actions based on its current state, receives feedback from the environment in the form of rewards, and updates its knowledge or policy to improve future decision-making. This feedback loop, known as the reinforcement learning loop, allows the agent to gradually learn how to make better decisions by maximizing expected rewards. As shown in Figure 3.1,

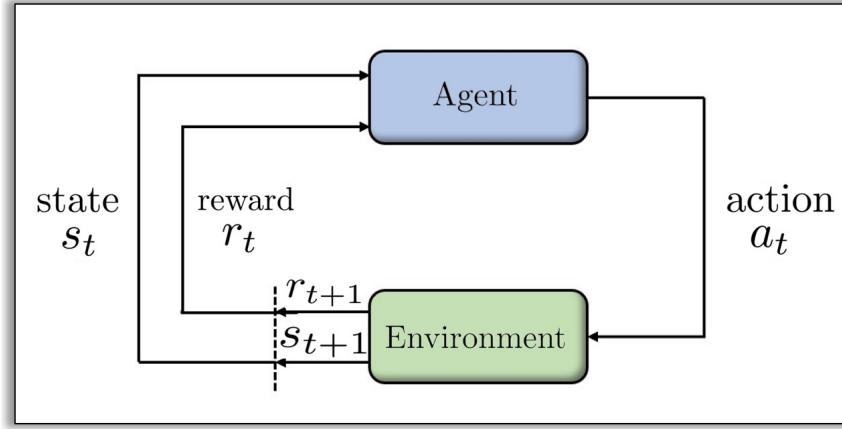


Figure 3.1: Reinforcement Learning Framework

the RL agent interacts with an environment, an environment in an RL setting produces a state s_t , and in response to state s_t , the agent takes an appropriate action a_t and moves to the next state s_{t+1} . For the action a_t in the state s_t , the agent receives a reward r_t from the environment. This quadruple (s_t, a_t, r_t, s_{t+1}) constitutes an *experience*. The agent collects many such experiences, and based on the reward associated with the experience, it determines the optimal behaviour or *policy*. A policy $\pi(s, a)$ is a mapping from each state to an action that determines how the agent acts at each state. An RL algorithm like *Q-learning* helps develop an optimal policy based on such a collection of experiences. The process of using gathered experience to determine an optimal policy is what constitutes training. After learning, the agent is supposed to choose the best action from any given state, where the best action is the one that maximizes long-term output. Below we describe the elements of RL in more detail:

- **Agent:** Agent is the key element in the RL. It is the entity that interacts with the environment and makes decisions. The agent receives observations from the environment, takes actions, and receives feedback in the form of rewards or punishments, which is used to train the agent.

- **Environment:** An environment is an external system with which the agent interacts. It could be a physical environment, a virtual simulation, or any other context in which the agent operates. The environment presents its current state to the agent, based on which the agent takes action.
- **State:** A state is the latest snapshot of the environment that represents the current situation or context in which the agent finds itself. It encapsulates relevant information for the agent to make decisions.
- **Action:** An action is a decision or behaviour chosen by the agent in response to a given environment state. Actions can have short-term consequences that affect the subsequent states and rewards.
- **Reward:** A Reward is a scalar feedback signal that indicates the desirability of an agent's action in a given state. Rewards can be positive, negative, or neutral and measure performance or goal achievement. The Rewards are used to train the agent in order to determine an optimal policy.
- **Policy:** A policy is the strategy or set of rules that the agent uses to determine its actions based on the current state of the environment. It maps states to actions and can be deterministic or stochastic. An optimal policy results in the highest cumulative reward for the agent.

Example 3.1.1: Example: Recycling Robot

In order to understand an RL task, let's take the example of a mobile robot whose job is to collect trash cans in an office environment. The robot is equipped with sensors to perceive its environment in order to detect cans and arms to pick up the cans and place them in a dustbin. The robot has a finite rechargeable battery. The robot's control system has components to interpret and process sensory information for controlling the arm and navigating. The goal of the robot is to collect cans

without getting discharged on its way to collect cans. Thus, depending on the battery level, the robot has to decide whether (i) to look for a can actively, (ii) to head back to the charging station to get recharged, or (iii) to remain idle. The above decision needs to be made periodically or when a certain event occurs. This problem can be modeled as an RL task in the following way,

- ***State Space:*** The robot's battery level determines the state of the robot.
- ***Action Space:*** The robot has three actions to choose from, (1) collect cans, (2) get recharged, or (3) remain idle.
- ***Reward:*** The rewards can be zero most of the time when searching for a can, or it is idle, but then becomes positive when the robot picks an empty can and negative if the robot gets discharged.
- ***Agent:*** The agent is not the entire robot, but only the system that monitors the robot's internal state (i.e., battery) and controls arms.
- ***Environment:*** The robot's external environment and the rest of the robot may include other complex systems.
- ***Policy:*** The RL algorithm learns a policy, represented as a mapping from states to actions, to decide on an action based on the current battery level.

Reinforcement learning algorithms utilize various techniques to estimate the value or utility of different state-action pairs and improve the agent's decision-making over time. These techniques include ***value-based methods*** (e.g., *Q-learning*), ***policy-based methods*** (e.g., *REINFORCE*), and ***actor-critic methods*** (e.g., *DDPG*). We present some of the important algorithms in this chapter. The algorithms discussed in this chap-

ter are taken from the *Reinforcement Learning: An Introduction* book by Sutton and Barto [42].

3.2 Prelimanaries and Definitions

Let S be the set of all possible states in the environment and s_t be one of the states from S at time t , $A(s_t)$ be the set of all possible actions in state s_t , let a_t be one of the actions from $A(s_t)$ at time t . R_t be the reward at time t and G_t be the cumulative reward starting from time t . RL agent aims to maximize the cumulative reward over T trials. Let $R_{t+1}, R_{t+2}, R_{t+3}, \dots, R_T$ are the sequence of rewards agent receives starting from time t . The cumulative rewards (or *return*) can be written as,

- *Undiscounted Return:*

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \dots + R_T$$

- *Discounted Return:*

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Here, $\gamma \in [0, 1]$ represents the discount factor that captures the future uncertainties; If $\gamma = 0$, agent becomes myopic and $\gamma = 1$ makes the agent far-sighted. Having a $\gamma < 1$ is a mathematical trick to get bounded returns in an infinite horizon problem setting.

Another important property is *Markov Property* [43], which says at each time t , an agent chooses an action that only depends on the current state s_t , not on the history. RL setting satisfies the Markov property if the reward R_{t+1} , and the next state s_{t+1} depend only on the current state s_t and the action a_t at the current state. Mathematically,

$$Pr\{R_{t+1} = r, S_{t+1} = s' | s_0, a_0, R_1, \dots, s_t, a_t\} = Pr\{R_{t+1} = r, S_{t+1} = s' | s_t, a_t\}$$

An RL task that satisfies the Markov property is called a ***Markov Decision Process***, which is defined below. An MDP with finite state and action spaces is called a Finite MDP.

Definition 3.2.1: Markov Decision Process (MDP)

A ***Markov Decision Process (MDP)*** [44, 45] is a Reinforcement Learning framework, which is a tuple represented by $M = (S, A, P, r, \gamma)$, where S denotes the set of states, A denotes the set of actions, P is the transition probability function, where $P(s' | s, a) = P(s_{t+1} = s' | s_t = s, a_t = a)$ is the probability that taking action a in state s at time t will lead to state s' at time $t+1$, r denotes the reward function, where $r(s, a)$ is the reward obtained by taking action a in state s , and $\gamma \in [0, 1]$ denotes the discount factor.

Below are some of the important definitions:

Definition 3.2.2: Policy

A ***policy*** π is a mapping from states to actions.

- ***Deterministic Policy***: A deterministic policy is represented by $\pi : S \rightarrow A$. The ***deterministic policy*** function maps each state $s \in S$ to a single action $a \in A$.
- ***Stochastic Policy***: A stochastic policy is represented by $\pi : S \times A \rightarrow [0, 1]$. A ***stochastic policy*** is a policy that maps each state $s \in S$ to a probability distribution over actions ΔA .

Definition 3.2.3: State-Value Function

The **state-value function** gives an expected return when an agent follows policy π starting from state s . For each state $s \in S$, its value under policy π is defined as

$$v_\pi(s) = \mathbb{E}_\pi[G_0|S_0 = s] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s \right]$$

The state-value of the state is dependent on the policy. The state-value function provides a way to compare different policies.

Definition 3.2.4: Action-Value Function

The **action-value function** gives an expected return when an agent starts from state s by taking action a and then follows policy π . For each state $s \in S$ and action $a \in A$, its value under policy π is defined as

$$q_\pi(s, a) = \mathbb{E}_\pi[G_0|S_0 = s, A_0 = a] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s, A_t = a \right]$$

Similar to the state-value function, the action-value of a state-action pair is dependent on the policy and provides a way to compare different policies.

Definition 3.2.5: Optimal State-Value Function

The **optimal state-value function**, denoted by v^* , is defined as

$$v^*(s) = \max_\pi v_\pi(s), \forall s$$

Definition 3.2.6: Optimal Action-Value Function

The **optimal action-value function**, denoted by q^* , is defined as

$$q^*(s, a) = \max_\pi q_\pi(s, a), \forall s \in S, \forall a \in A(s)$$

Definition 3.2.7: Optimal Policy

Let π^* be an *optimal policy*, then

$$\pi^* = \operatorname{argmax}_{\pi} v_{\pi}(s), \forall s \in S$$

or

$$\pi^* = \operatorname{argmax}_{\pi} q_{\pi}(s, a), \forall s \in S, \forall a \in A(s)$$

Bellman Expectation Equations

$$v_{\pi}(s) = \sum_{a \in A(s)} \pi(a|s) \left[r(s, a) + \gamma \sum_{s' \in S} p(s'|s, a) v_{\pi}(s') \right] \quad (3.1)$$

$$q_{\pi}(s, a) = r(s, a) + \gamma \sum_{s' \in S} p(s'|s, a) v_{\pi}(s') \quad (3.2)$$

Bellman Optimality Equations

$$v^*(s) = \max_a \left[r(s, a) + \gamma \sum_{s' \in S} p(s'|s, a) v_{\pi}(s') \right] \quad (3.3)$$

$$q^*(s, a) = r(s, a) + \gamma \sum_{s' \in S} p(s'|s, a) \max_{a'} q^*(s', a') \quad (3.4)$$

3.3 Tabular RL Methods

Tabular methods are methods where the state and action spaces are small enough to be represented in the form of tables for the value function approximation. The MDPs, in the case of tabular RL methods, are known as finite MDPs. The next subsections 3.3.1, 3.3.2 and 3.3.3 describe the fundamental classes of solving finite MDP problems, namely, *dynamic programming (DP)*, *Monte Carlo methods* and *temporal difference (TD)* methods. Figure 3.2 shows the fundamental difference between the three classes of algorithms.

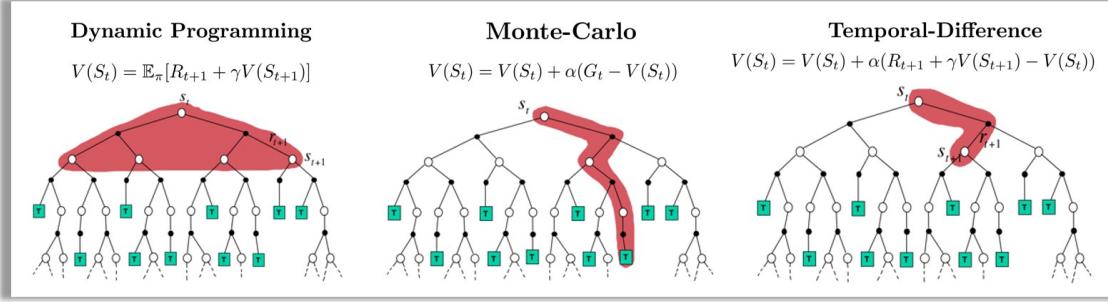


Figure 3.2: Comparison of the update methods of Dynamic Programming (DP), Monte Carlo (MC) and Temporal Difference (TD) learning for state value functions [Image Credit: © [3]]

3.3.1 Dynamic Programming

Dynamic Programming (DP) [46] is a fundamental approach in Reinforcement Learning (RL) that involves breaking down a complex problem into smaller, more manageable subproblems. It uses the principle of optimality to solve the subproblems and iteratively builds up the optimal solution. DP requires complete knowledge of the environment dynamics, including the transition probabilities and rewards, which makes it suitable for solving RL problems with a known model. We usually assume that the environment is a finite MDP with states S , action $A(s)$, $\forall s \in S$ are finite, and that its dynamics are given by a set of probabilities $p(s', r|s, a)$, $\forall s, s' \in S, a \in A(s)$.

3.3.1.1 Policy Iteration

Policy iteration is an algorithm used in RL to find an optimal policy for an MDP [46, 47]. It is an iterative method that combines two steps: **policy evaluation** and **policy improvement**. Algorithm 1 shows the working of the policy iteration method.

We start the algorithm by initializing a random or arbitrary policy.

Policy Evaluation: Evaluate the value function for the current policy. This involves estimating the expected cumulative reward for each state under the current policy. The value function can be iteratively updated using techniques like iterative policy evaluation or solving the Bellman equation.

Policy Improvement: Once the value function has been evaluated, improve the policy by selecting the action that maximizes the expected cumulative reward from each state. This is done by updating the policy to be greedy with respect to the value function and selecting the action with the highest estimated value.

The method iterates between policy evaluation and policy improvement until the policy converges to an optimal policy, where no further changes in the policy occur. The policy iteration algorithm guarantees convergence to an optimal policy as long as the MDP is finite and the policy evaluation step accurately estimates the value function. At each iteration, the policy is improved based on the current value function, which is then updated based on the improved policy. This process continues until the policy becomes optimal. Policy iteration provides a systematic approach to finding the optimal policy by iteratively refining both the policy and the value function. It exploits the interplay between policy evaluation and policy improvement to converge to an optimal solution.

While policy iteration is a robust algorithm, it can be computationally expensive, especially for large state and action spaces. Each iteration requires solving a set of linear equations or optimization problems, which can be time-consuming. Therefore, for more complex problems, approximate methods such as value iteration and Q-learning are often used as they can be more efficient. Overall, policy iteration is a foundational algorithm in reinforcement learning and provides a clear framework for finding optimal policies. It serves as a building block for more advanced algorithms.

Policy Iteration

Algorithm 1: Policy_Iteration()

Input: None

Output: Value function $V()$ and Policy $\pi()$

1: Initialize $V(s) \in \mathbb{R}$, and $\pi(s) \in A(s)$ arbitrary $\forall s \in S$

Policy Evaluation

2: $\Delta \leftarrow 0$, initialize θ with a small value

3: **while** $\Delta > \theta$ **do**

4: **for** each $s \in S$ **do**

5: $v \leftarrow V(s)$

6: $V(s) \leftarrow \sum_{s',r} p(s',r|s,\pi_s)[r + \gamma V(s')]$

7: $\Delta \leftarrow \max(\Delta, |v - V(s)|)$

8: **end for**

9: **end while**

Policy Improvement

10: $policy_stable \leftarrow true$

11: **for** each $s \in S$ **do**

12: $oldAction \leftarrow \pi(s)$

13: $\pi(s) \leftarrow argmax_a \sum_{s',r} p(s',r|s,a)[r + \gamma V(s')]$

14: If $old_Action \neq \pi(s)$, then $policy_stable \leftarrow false$

15: **end for**

16: **if** $policy_stable$ **then**

17: stop and **return** $V \approx v_*$ and $\pi \approx \pi_*$

18: **else**

19: go to **Policy Evaluation**

20: **end if**

3.3.1.2 Value Iteration

The **value iteration** algorithm is another iterative method used in RL to find the optimal value function and policy for an MDP [48]. Unlike the policy iteration method, it combines policy evaluation and policy improvement into a single step. Algorithm 2 shows the working of the algorithm.

We start the method by initializing the value function for each state to arbitrary values or zeros. Then, for each state in the MDP, calculate the updated value using the Bellman optimality equation, as shown in Step 6. $V(s)$ represents the value function for state s . Repeat this process until the value function converges (i.e., until the change in the value function from one iteration to the next becomes small, indicating convergence). Finally, once the value function has converged, extract the optimal policy by selecting the action with the highest Q-value for each state, as shown in Step 10. $\pi(s)$ represents the optimal policy for state s .

The value iteration algorithm iteratively updates the value function by repeatedly applying the Bellman optimality equation. At each iteration, the value function is refined, leading to a more accurate estimation of the expected cumulative reward for each state. The algorithm continues until the value function converges, guaranteeing convergence to the optimal value function. After convergence, the optimal policy can be extracted by selecting the action with the highest Q-value for each state. The resulting policy represents the best action to take in each state to maximize the expected cumulative reward.

Value iteration is computationally efficient as it combines both policy evaluation and policy improvement in a single step. However, it may require a large number of iterations to converge, especially for problems with large state spaces.

Value Iteration

Algorithm 2: Value Iteration()

Input: None

Output: Policy $\pi()$

```

1: Initialize  $V(s)$  arbitrarily  $\forall s \in S$  except terminal,  $V(\text{terminal}) \leftarrow 0$ 
2:  $\Delta \leftarrow 0$ , initialize  $\theta$  with a small value
3: while  $\Delta > \theta$  do
4:   for each  $s \in S$  do
5:      $v \leftarrow V(s)$ 
6:      $V(s) \leftarrow \max_a \sum_{s',r} p(s',r|s,a)[r + \gamma V(s')]$ 
7:      $\Delta \leftarrow \max(\Delta, |v - V(s)|)$ 
8:   end for
9: end while
10: return  $\pi$  s.t.  $\pi(s) = \arg\max_a \sum_{s',r} p(s',r|s,a)[r + \gamma V(s')]$ 

```

3.3.2 Monte Carlo

The **Monte Carlo method** is a sampling-based approach in RL that learns from experience by directly interacting with the environment. It collects sample trajectories by executing a policy and uses these trajectories to estimate the value function or find the optimal policy. Monte Carlo methods do not require a known model of the environment and can be applied to both episodic and continuous tasks. The basic idea behind Monte Carlo algorithms is to sample episodes by interacting with the environment and collecting sequences of states, actions, and rewards. These episodes are then used to estimate the value function or policy by averaging the observed returns.

There are many variants of the Monte Carlo algorithm. For instance, **Monte Carlo First Visit** [49] averages the returns for the first time state s visited in the episode;

Monte Carlo Every Visit [50] takes the average every time state s visited in the episode; On Policy Monte Carlo Control that evaluates and improves the same policy which is being used for the exploration/sampling; **Off Policy Monte Carlo Control** where one policy is used for exploration (called behaviour policy) and other policy is being leaned (called target policy). Both on-policy and off-policy can be implemented with either the first visit or every visit algorithm. Below we explain the Monte Carlo First Visit algorithm.

Monte Carlo First Visit algorithm ensures that only the first visit to a state in an episode is considered when updating the value function. This variant is particularly useful when dealing with episodic tasks or environments where the agent's interactions are divided into discrete episodes.

Algorithm 3 shows the working of the algorithm. The algorithm interacts with the environment by following a policy and generating episodes. An episode consists of a sequence of states, actions, and rewards, starting from an initial state until a terminal state is reached. For each episode, keep track of the states that are visited for the first time. For each visited state in the episode, calculate the return, which is the cumulative sum of rewards from that state until the end of the episode. After all episodes are completed, update the value function by averaging the returns for each visited state. Only the first visit to a state in an episode contributes to the update. Iterate over multiple episodes, updating the value function based on the first visits to the states. By considering only the first visit to each state, the Monte Carlo First Visit algorithm avoids potential biases that may arise from multiple visits to the same state within an episode. This ensures an unbiased estimation of the value function or policy.

Monte Carlo algorithms have several advantages, including their ability to learn directly from sampled experiences without requiring a model of the environment. However, they can be computationally expensive and may require a large number of episodes to converge to accurate estimates, especially for problems with large state spaces. Monte Carlo methods

and their variants have been widely applied in RL, particularly in scenarios where model-free learning is necessary or when the dynamics of the environment are unknown.

First Visit Monte Carlo

Algorithm 3: First_Visit_Monte_Carlo(π)

Input: Policy $\pi()$

Output: None (just update $Q(s, a)$)

```

1: Initialize  $Q(s, a) \leftarrow$  arbitrary,  $C(s, a) \leftarrow 0, \forall s \in S, \forall a \in A(s)$ 
2: while true do
3:    $b \leftarrow$  any policy with coverage of  $\pi$ 
4:   Generate an episode using  $b$ :  $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T, S_T$ 
5:    $G \leftarrow 0, W \leftarrow 1$ 
6:   for  $t = T - 1, T - 2, \dots, 0$  do
7:      $G \leftarrow \gamma G + R_{t+1}$ 
8:      $C(S_t, A_t) \leftarrow C(S_t, A_t) + W$ 
9:      $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)}[G - Q(S_t, A_t)]$ 
10:     $W \leftarrow W \frac{\pi(A_t | S_t)}{b(A_t | S_t)}$ , If  $W = 0$  then exit For loop
11:   end for
12: end while

```

3.3.3 Temporal-Difference Learning

Temporal Difference (TD) methods combine ideas from dynamic programming and Monte Carlo methods. TD methods learn directly from incomplete sequences of experience. They update value estimates based on the observed reward and the estimate of the value of the next state. TD methods utilize a bootstrapping approach, where they update their value estimates based on other value estimates. TD methods are model-free and can learn

from incomplete and ongoing interactions with the environment. Below, we explain two of the widely used TD methods, namely, **SARSA** and ***Q*-learning**.

3.3.3.1 SARSA: On-Policy TD Control

SARSA is an acronym for ***State-Action-Reward-State-Action***, which is an algorithm used in RL to learn an optimal policy for decision-making in an environment [51, 52]. It is an on-policy TD algorithm that updates the Q-values based on the current state, action, reward, and the next state and action. The algorithm 4 shows the working Q-Learning algorithm.

As shown in the algorithm, we start by initializing the Q-values for all state-action pairs to arbitrary values or zeros. Then, we initialize a state s_t . In each time step, the agent selects an action a_t in state s_t based on an exploration-exploitation strategy. This can be an ϵ -greedy strategy, where the agent chooses the action with the highest Q-value with probability $1 - \epsilon$ (exploitation) or selects a random action with probability ϵ (exploration). After taking action a_t , the agent observes the reward r_t from the environment and moves to the next state s_{t+1} . The agent selects the next action a_{t+1} based on the next state s_{t+1} using the exploration-exploitation strategy. Then, using the observed reward r_t , the next state s_{t+1} , and the next action a_{t+1} , the agent updates the Q-value of the current state-action pair (s_t, a_t) using the formula shown in Step-7. In the formula, $Q(s_t, a_t)$ denotes the Q-value of state-action pair (s_t, a_t) , $\alpha \sim [0, 1]$ is the learning rate, which determines the extent to which the Q-values are updated, and $\gamma \sim [0, 1]$ is the discount factor that determines the importance of future rewards. It controls the balance between immediate and long-term rewards. We run the for loop until the termination condition is met.

During the learning process, the agent interacts with the environment, updates the Q-values based on the observed rewards and the Q-values of the next state-action pair, and gradually improves its policy. The Q-values converge to their optimal values by repeatedly applying the SARSA algorithm, representing the maximum expected cumulative rewards

for each state-action pair. The SARSA algorithm ensures that the learned policy is on-policy, meaning that it takes into account the exploration strategy used during learning. This makes SARSA well-suited for tasks where the learned policy must account for the consequences of the agent's actions.

SARSA

Algorithm 4: SARSA()

Input: None

Output: None (just update $Q(s, a)$)

```

1: Initialize  $Q(s, a) \forall s \in S, \forall a \in A(s)$ 
2: for  $t = 1$  to  $T$  do
3:   Sample an initial state  $s_t$ 
4:   while  $s_t \neq$  terminal state do
5:     Choose action  $a_t$  in state  $s_t$  from the policy derived from  $Q$  (using
 $\epsilon$ -greedy)
6:     Perform action  $a_t$  and observe reward  $r_t$  and next state  $s_{t+1}$ 
7:      $Q(s_t, a_t) = (1 - \alpha) * Q(s_t, a_t) + \alpha * [r_t + \gamma * Q(s_{t+1}, a_{t+1})]$ 
8:      $s_t \leftarrow s_{t+1}$ 
9:   end while
10: end for
```

SARSA can be applied to a wide range of RL problems and has been successfully used in various domains, including robotics, game-playing, and control systems. SARSA enables the agent to learn an optimal policy for decision-making in uncertain and dynamic environments by iteratively updating the Q-values based on observed rewards and taking action accordingly.

3.3.3.2 Q-Learning: Off-Policy TD Control

Q-learning, similar to SARSA, is a popular algorithm in RL used to learn an optimal policy for decision-making in an environment [53, 54]. The key difference between the two algorithms is in the update step; SARSA uses the agent's actual next action (thus, **on-policy**), while Q-learning uses the action with the maximum Q-value in the next state (thus, **off-policy**). So, Q-learning is a model-free, off-policy algorithm that is well-suited for solving MDPs, where an agent interacts with an environment and learns to take actions that maximize its cumulative rewards. In Q-learning, the algorithm aims to determine the Q-value function for each state-action pair, also known as the action-value function. The Q-value of a state-action pair (s, a) represents the expected cumulative reward the agent will receive by taking action a in state s and following the optimal policy thereafter. The algorithm 5 shows the working Q-Learning algorithm.

As described in the algorithm, we start by initializing the Q-values for all state-action pairs. This can be done randomly or with some initial assumptions. In each time step, the agent selects an action a_t in state s_t based on an exploration-exploitation strategy. This can be an ϵ -greedy strategy, where the agent chooses the action with the highest Q-value with probability $1 - \epsilon$ (exploitation) or selects a random action with probability ϵ (exploration). After taking action a_t , the agent observes the reward r_t from the environment and transitions to the next state s_{t+1} . Then, using the observed reward r_t and the next state s_{t+1} , the agent updates the Q-value of the current state-action pair using the formula shown in Step 7. The notations in the formula (α and γ) are similar to the above-presented SARSA algorithm. The agent continues to select actions, observe rewards, update Q-values, and transition to the next state until the termination condition is met (e.g., reaching a goal state or a certain number of iterations).

Over time, the Q-values converge to their optimal values, representing the maximum expected cumulative rewards for each state-action pair. The agent can then use the learned Q-values to determine the optimal policy by selecting the action with the highest Q-value

for each state. Q-learning is a powerful algorithm that can learn optimal policies in complex RL problems without prior knowledge of the environment dynamics. It has been widely applied in various domains, such as robotics, game-playing, and autonomous systems, to enable intelligent decision-making in uncertain environments.

Q-Learning

Algorithm 5: Q_Learning()

Input: None

Output: None (just update $Q(s, a)$)

```

1: Initialize  $Q(s, a) \forall s \in S, \forall a \in A(s)$ 
2: for  $t = 1$  to  $T$  do
3:   Sample an initial state  $s_t$ 
4:   while  $s_t$  != terminal state do
5:     Choose action  $a_t$  in state  $s_t$  from the policy derived from  $Q$  (using
 $\epsilon$ -greedy)
6:     Perform action  $a_t$  and observe reward  $r_t$  and next state  $s_{t+1}$ 
7:      $Q(s_t, a_t) = (1 - \alpha) * Q(s_t, a_t) + \alpha * [r_t + \gamma * (\max_{a_{t+1}} Q(s_{t+1}, a_{t+1}))]$ 
8:      $s_t \leftarrow s_{t+1}$ 
9:   end while
10: end for
```

3.4 Approximate Solution Methods

In many of the tasks or real-world scenarios to which we would like to apply RL, the *state space is combinatorial and enormous*, or the *action space is continuous*. In such cases, the tabular methods discussed above would not be helpful as we cannot expect to find an optimal policy or the optimal value function even in the limit of infinite

time and data. Instead, in such situations, our goal is to find a good approximate solution using limited computational resources. The difficulty with large state spaces is not just about the storage requirements to save large tables but also the time and data needed to fill them accurately. It is common to encounter a never-seen state during the test phase. In order to make sensible decisions in such states, the algorithms should be such that they can generalize from previous encounters with different states that are in some sense similar to the current one. Below we present a popular class of methods used for approximation, known as the policy gradient methods.

3.4.1 Policy Gradient Methods

Policy gradient methods are a class of RL algorithms that directly optimize the policy to maximize the expected cumulative reward. Unlike value-based methods that estimate the value function, policy gradient methods focus on learning the policy itself. In policy gradient methods, the policy is typically represented by a parameterized function, such as a neural network, that takes the state as input and outputs a probability distribution over the action space. The objective is to find the policy parameters that maximize the expected return. The basic idea behind policy gradient methods is to update the policy parameters in the direction of the gradient of an objective function, often referred to as the policy gradient. The objective function is typically formulated as the expected cumulative reward, weighted by a factor called the advantage function, which represents how much better or worse an action is compared to the average. They can handle *both discrete and continuous action spaces*, and they can *learn stochastic policies* that capture exploration. Policy gradient methods have become a prominent approach in RL due to their ability to directly optimize policies, making them well-suited for problems with high-dimensional or continuous action spaces and tasks that require exploration and adaptation to changing environments. Below, we present one of the successful policy gradient methods, DDPG.

3.4.1.1 Deep Deterministic Policy Gradient (DDPG)

Deep Deterministic Policy Gradient [55], commonly known as **DDPG**, is an actor-critic, model-free reinforcement learning (RL) algorithm based on the concept of deterministic policy gradients [56] that combines elements of both policy-based and value-based methods. This algorithm works on continuous action spaces and has been proven to be very successful in a variety of continuous control problems [55], e.g., TORCS and MUJOCO environments.

DDPG is an off-policy algorithm that learns a *deterministic policy* (as opposed to a stochastic policy) and uses a combination of deep neural networks to approximate both the policy and the value function. The algorithm leverages the strengths of Deep Q-Networks (DQN) and deterministic policy gradient methods to handle continuous action spaces. The algorithm deploys a parameterized policy (actor) and critic network that gets updated frequently using experiences gathered from sample roll-outs. While the policy network map learns an optimal deterministic mapping from state space to action space, the critic network learns the optimal value function using Bellman-type updates used in Q-learning. Figure 3.3 shows the structural framework of the DDPG algorithm.

Algorithm 6 shows the working of the DDPG algorithm. As per the algorithm, start with initializing the actor and critic networks. The actor-network approximates the policy and takes the state as input to output the action. The critic network approximates the action-value function (Q-function) and takes both the state and action as inputs to output the estimated Q-value. Then, the target networks for both actor and critic will be initialized by making copies of the original networks. These target networks are used to compute the target Q-values to update the networks. Create a replay buffer to store the agent's experiences, including the state, action, reward, next state, and done flag. Then, for each episode, initialize a random process to generate noise, which is used for exploration. Initialize the state s_t and interact with the environment by selecting action a_t based on the actor's current policy (exploration-exploitation strategy) and collect experiences. Then, observe

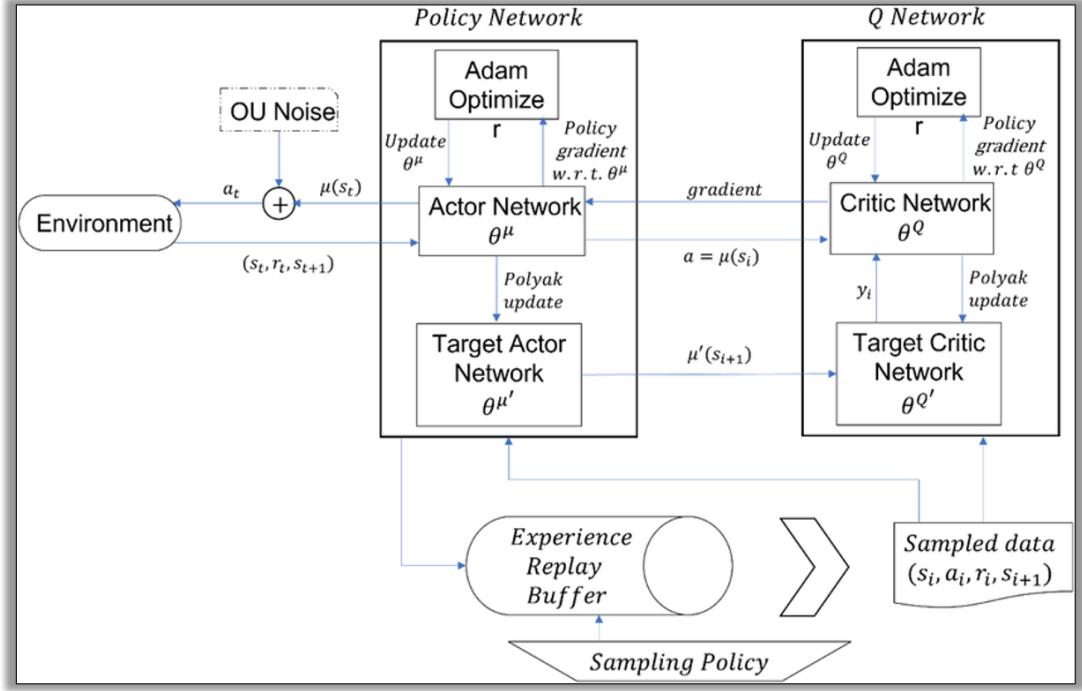


Figure 3.3: The DDPG Algorithm Structure Framework [Image Credit: © [4]]

the reward r_t and the next state s_{t+1} ; after that store the transition tuple (s_t, a_t, r_t, s_{t+1}) to replay buffer. To train the networks, sample a mini-batch of experiences from the replay buffer and perform Steps 12 to 15. That is, compute the target Q-values using the Bellman equation with the use of target actor and target critic networks. Then, update the critic network by minimizing the mean squared error between the predicted Q-values and the target Q-values. After that, the actor-network will be updated using the sampled gradient from the critic network to maximize the estimated Q-value. Finally, update the target networks by slowly blending the original networks with the current network weights using a soft update. Repeat the process until the termination condition is met.

DDPG leverages the insights from Deep Q-Learning by using an off-policy approach and experience replay to stabilize learning. The use of target networks helps to improve the stability of training by providing more consistent target values. Based on the Bellman

equation, the algorithm uses gradient descent to update the actor and critic networks. Its ability to handle continuous action spaces, combined with the power of deep networks, makes it a popular choice for RL tasks requiring precise control in continuous domains.

DDPG

Algorithm 6: DDPG()

Input: None

Output: None (just update critic-network $Q()$ and actor-network $\mu()$)

- 1: Randomly initialize critic-network $Q(s, a|\theta^Q)$ and actor-network $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ , Initialize replay buffer R
 - 2: Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
 - 3: **for** $episode = 1$ to M **do**
 - 4: Initialize a random process \mathcal{N} for action exploration (basically to add noise)
 - 5: Receive initial observation state s_t
 - 6: **for** $t = 1$ to T **do**
 - 7: Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to current policy and noise
 - 8: Execute action a_t and observe reward r_t and next state s_{t+1}
 - 9: Store transition tuple (s_t, a_t, r_t, s_{t+1}) in R
 - 10: Sample a minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
 - 11: Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
 - 12: Update critic by minimizing the loss: $L = \frac{1}{N} \sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
 - 13: Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta_\mu} \mu(s|\theta^\mu)|_{s_i}$$
 - 14: Update the target networks:

$$\theta^{Q'} \leftarrow \tau \theta^Q + (1 - \tau) \theta^{Q'}, \theta^{\mu'} \leftarrow \tau \theta^\mu + (1 - \tau) \theta^{\mu'}$$
 - 15: **end for**
 - 16: **end for**
-

3.5 Monte-Carlo Tree Search: An Overview

Having seen the policy gradient methods that optimize policies through direct gradient-based updates, we now explore **Monte Carlo Tree Search (MCTS)**, an approach that combines planning and simulation to guide decision-making in complex environments. MCTS is a powerful algorithm in the realm of artificial intelligence and computer science, particularly acclaimed for its effectiveness in decision-making and problem-solving in complex, strategic environments. Originally developed for game-playing scenarios, MCTS has since found applications in various domains, from robotics to optimization. It combines the principles of **tree search with random sampling** to make optimal decisions. Unlike traditional tree search methods, MCTS employs a stochastic sampling approach to traverse the search space, effectively balancing exploration and exploitation. MCTS constructs a search tree incrementally, starting from the current game state, and iteratively simulates potential future actions by using random playouts to evaluate different paths. By simulating numerous random playouts from each node of a decision tree and updating statistics accordingly, MCTS gradually refines its understanding of the problem landscape, ultimately guiding toward optimal or near-optimal solutions. The process consists of four main steps: **selection**, **expansion**, **simulation**, and **backpropagation**.

- **Selection:** The algorithm traverses the current tree by selecting child nodes based on a balancing metric, such as the Upper Confidence Bound (UCB), until it reaches a node that has not been fully explored.
- **Expansion:** If the selected node is not a terminal state, one or more child nodes are added to expand the tree.
- **Simulation:** A simulation (or playout) is run from the newly added node to estimate the outcome using random or heuristic-based moves until a terminal state is reached.

- **Backpropagation:** The result of the simulation is propagated back up the tree, updating the statistics for each node along the path, including visit counts and value estimates.

MCTS has been successfully applied in various domains, such as board games like Go and chess, due to its ability to balance exploration and exploitation effectively. The algorithm's strength lies in its flexibility and capacity to handle vast search spaces without needing a full game tree expansion, making it suitable for complex, high-dimensional problems. MCTS has been traditionally applied in problems where the actions are discrete, for example, Chess and Go; however, recent work has shown that MCTS can be effectively applied to continuous actions as well. Below, we briefly talk about both versions of MCTS.

3.5.1 MCTS for Discrete Action Spaces

MCTS is a simulation-based search approach to planning in finite-horizon sequential decision-making settings. The core of the approach is to iteratively simulate executions from the current state to a terminal state, incrementally growing a tree of simulated states (nodes) and actions (edges). Each simulation starts by visiting nodes in the tree and selecting which actions to take based on a *selection function* and information maintained in the node. Consequently, it transitions to a successor state. When it encounters a node that is not fully expanded, then the node is *expanded* by adding a new leaf to the tree. Then, a *simulation* is performed from the new leaf to a terminal state. The value of the terminal state is then returned as the value for that new leaf, and the information stored in the tree is updated via a *backpropagation* step.

The most widely used selection function for MCTS is *Upper Confidence Bounds Applied to Trees (UCT)* [57]. In UCT, each node maintains the mean of the rewards received for each action, \bar{v}_a , and the number of times each action has been selected, n_a . $\sum_b n_b$ is the total number of times the node is visited. It initially tries each action once

and then chooses the next action based on the size of the one-sided confidence interval on the reward, computed using the Chernoff-Hoeffding bound as shown below,

$$\operatorname{argmax}_a \left[\bar{v}_a + C \sqrt{\frac{\log \sum_b n_b}{n_a}} \right] \quad (3.5)$$

This bound is controlled by the constant C , which determines the exploration-exploitation trade-off and is typically fine-tuned for the specific domain. However, one of the primary limitations of discrete MCTS methods is the fixed size of the action space, which is addressed by progressive widening methods.

3.5.2 MCTS for Continuous Action Spaces: Simple Progressive Widening (SPW)

In *continuous MCTS*, the vanilla UCT no longer works since each action should be tried at least once, and there are infinitely many actions to be considered. *Progressive widening* addresses this challenge by maintaining a finite list of actions to be explored and incrementally adding new child action nodes v_c to this list based on visitation counts. Specifically, a new child node is added whenever the following condition is met:

$$n(v_c)^\alpha \geq |\text{children}(v_c)|$$

Here, $\alpha \in (0, 1)$ is a parameter controlling the growth rate, $n(v_c)$ is the visit count of the node v_c 's and $|\text{children}(v_c)|$ is the size of node v_c children list. This ensures that before adding a new action to the list, the current set of actions gets visited sufficient times. When a new node v_d is created, a new action particle is either sampled from a probability distribution $a \sim \pi_{\text{sampler}}(|s|)$ or deterministically generated (e.g., to expand the action space coverage). This generated action particle is stored in $\text{action}(v_d)$, and the process iterates accordingly.

While RL addresses sequential decision-making problems with delayed rewards, we now turn our attention to multi-armed bandits (MAB), a simpler yet fundamental framework that focuses on balancing exploration and exploitation in single-step decision scenarios.

3.6 Multi-Armed Bandits and its Derivatives: An Overview

Multi-armed bandits (MAB) are a class of sequential decision-making problems that involve a trade-off between exploration and exploitation. The name **bandit** comes from the idea of a gambler pulling the arm of a slot machine (a one-armed bandit) to maximize their winnings over time. In the context of multi-armed bandits, the **arms** represent different actions or options that an agent can choose from, and the agent's goal is to maximize its cumulative rewards by selecting the best arm(s) over a series of trials.

In an MAB problem, the agent faces a set of arms, each associated with an unknown reward distribution. The agent does not initially know which arm(s) yield the highest rewards and must explore different arms to gather information. At each time step, the agent selects an arm, receives a reward from the chosen arm's distribution, and updates its knowledge based on the observed reward. The agent's objective is to find a balance between exploring unexplored arms to gather information and exploiting arms that are believed to yield higher rewards based on current knowledge.

There are two main strategies in MABs:

- **Exploration:** The agent explores arms to gain information about their reward distributions. This can involve randomly selecting arms or using specific exploration algorithms to balance exploration across arms. By exploring, the agent aims to reduce uncertainty about the rewards associated with different arms.
- **Exploitation:** The agent exploits the arms that have been identified as potentially high-reward options based on current knowledge. This involves selecting arms that are believed to yield higher rewards based on the available information. The agent

aims to maximize its rewards by exploiting the arms that have shown promising results so far.

The challenge in MABs is to strike the right balance between exploration and exploitation to maximize cumulative rewards. The agent must constantly update its beliefs about the arms' reward distributions as it receives feedback and learns from the observed rewards. Several algorithms, such as epsilon-greedy, upper confidence bound (UCB), and Thompson sampling, have been developed to address this exploration-exploitation trade-off and find effective arm selection strategies.

MABs have practical applications in various domains. For example, in online advertising, different ads can be seen as arms, and the goal is to identify the most effective ad to display to maximize user engagement. In clinical trials, different treatment options can be viewed as arms, and the objective is to determine the most effective treatment with the highest patient outcomes. MAB algorithms provide a framework for optimizing decision-making under uncertainty and limited resources, enabling efficient learning and adaptive decision-making. MAB settings can vary based on the specific characteristics of the problem and the available information. Contextual Bandits, Non-Stationary Bandits, Budgeted Bandits, Collaborative Bandits, Bayesian Bandits and Adversarial Bandits are some of the widely studies MAB settings. In this work, we utilize concepts from Contextual MAB; thus, we describe Contextual MAB in detail.

3.6.1 Contextual MAB

Contextual multi-armed bandits (MAB) [58] extend the basic multi-armed bandit framework by incorporating additional context or state information as shown in Figure 3.4. In this setting, each arm is not only characterized by its action but also by a set of context features that provide additional information about the current state or context of the problem. Contextual MAB algorithms aim to learn a policy that maps the observed context to the most promising action, taking into account the context-action-reward relationship.

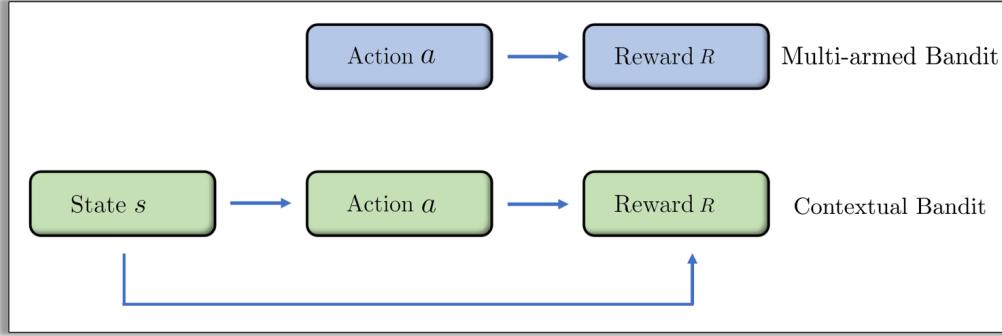


Figure 3.4: MAB vs. Contextual Bandits

Contextual MAB tasks are intermediate between the MAB problem and the full RL problem. They are like the full RL problem in that they involve learning a policy, but they are also like MAB in that each action affects only the immediate reward. If actions are allowed to affect the next state as well as the reward, then we have the RL problem. Below are the important elements of contextual MAB.

In the MAB algorithm, the output action does not depend on any state or context. However, In Contextual MAB, the algorithm makes personalized decisions based on each context. Thus, the algorithm observes a context, makes a decision based on the context (which is choosing one action from the set of actions), and observes a reward for that action. Here too, the goal is to maximize the average reward.

More formally, there is a set of context or feature vectors denoted as X . The context vectors describe the relevant information about the problem at hand, such as user attributes, time of day, or environmental conditions. There is a set of arms or actions denoted as A . The task is to find the best-suitable arm given the context. Below are some of the elements of Contextual MAB.

- **Action Selection:** At each time step t , given the current context x_t , the algorithm needs to select an action or arm a_t . The action selection strategy aims to maximize

the expected cumulative reward. The selection strategy can be based on the estimated reward distributions, the context features, or a combination of both.

- **Feedback and Reward:** After selecting an action a_t based on the context x_t , the algorithm receives feedback in the form of a reward r_t . The reward reflects the quality or desirability of the chosen action in the given context. The reward can be binary (e.g., click or no-click), continuous (e.g., revenue), or any other appropriate metric depending on the problem domain.
- **Reward Distribution:** Each context-action pair (x, a) has an associated unknown reward distribution, denoted as $R(x, a)$. The goal is to learn the reward distribution for each context-action pair using observed feedback.
- **Learning and Model Updates:** The algorithm updates its knowledge based on the observed rewards and context-action pairs. It aims to learn the reward distribution or the expected value of rewards for each context-action pair. The learning process involves updating the estimates or models of the reward distributions using techniques such as Bayesian inference, maximum likelihood estimation, or online learning algorithms.
- **Exploitation and Exploration:** Contextual MAB algorithms need to strike a balance between exploitation and exploration. Exploitation refers to selecting the context-action pair with the highest estimated reward based on the current knowledge. Exploration involves trying out context-action pairs that have not been extensively explored to gather more information and refine the estimates. The balance between exploration and exploitation is crucial to find the optimal policy.
- **Learning Algorithm:** Contextual MAB algorithms employ various learning algorithms to update the reward estimates and make action selections. These algorithms can include Thompson Sampling, LinUCB, Contextual Bandit Gradient, or variants

of the epsilon-greedy approach tailored for contextual information. The choice of algorithm depends on the specific problem, the available information, and the desired trade-off between exploration and exploitation.

Contextual multi-armed bandits are applicable in numerous domains, including personalized recommendations, online advertising, dynamic pricing, and healthcare treatment. By considering the context in decision-making, these algorithms enable more tailored and adaptive strategies, allowing the agent to learn and optimize its actions based on the context-specific information available.

3.6.2 MAB Algorithms

Several types of MAB algorithms have been developed to tackle different aspects of the exploration-exploitation trade-off. Below are some of the widely studied algorithms, namely, *Epsilon-Greedy*, *UCB* and *EXP-3*.

3.6.2.1 The Epsilon-Greedy Algorithm

The *epsilon-greedy* algorithm is a simple and widely used approach for solving the exploration-exploitation trade-off in multi-armed bandit problems [59, 60]. It provides a balance between exploring unknown arms and exploiting the arms with the highest estimated rewards. Algorithm 7 shows the working of the epsilon-greedy algorithm.

As shown in the Algorithm 7, we start by initializing the estimated reward values for each arm in A . Additionally, set the exploration parameter ϵ , which determines the probability of exploration. At each time step t , generate a random number n between 0 and 1. If this random number n is less than ϵ , choose a random arm uniformly at random (exploration). This allows for the exploration of less explored arms. Otherwise, select the arm with the highest estimated reward value (exploitation). After selecting an arm a_t and performing the associated action, receive a reward r_t . Update the estimated reward value for the chosen arm based on the observed reward. The update can be performed using techniques such

as sample averaging or incremental updates. As the algorithm progresses, the estimated reward values for the arms are continuously updated based on the observed rewards. This information is used to guide the balance between exploration and exploitation. Initially, exploration is favoured due to the higher ϵ value. However, exploitation becomes more dominant as the algorithm learns and the estimated reward values become more accurate.

Epsilon-Greedy

Algorithm 7: Epsilon_Greedy()

Input: None

Output: None (just update $w(t)$)

- 1: Initialize/Load $w[|A|]$
 - 2: Initialize ϵ
 - 3: Sample a uniform number n between 0 and 1
 - 4: **if** $n < \epsilon$ **then**
 - 5: $a_t \leftarrow$ random action from the action space A and observe reward r_t
 - 6: **else**
 - 7: $a_t \leftarrow \max(w)$ and observe reward r_t
 - 8: **end if**
 - 9: Update $w(t)$ based on r_t
 - 10: **return** selected action a_t
-

The epsilon-greedy algorithm allows for a trade-off between exploration and exploitation by adjusting the value of epsilon. A higher ϵ value encourages more exploration, ensuring that less-explored arms are given a chance to yield higher rewards. Conversely, a lower ϵ value emphasizes exploitation, focusing on the arms that are currently estimated to be the most rewarding.

3.6.2.2 The UCB Algorithm

The ***Upper Confidence Bound (UCB)*** algorithm is a popular approach for solving the exploration-exploitation trade-off in multi-armed bandit problems [61]. It balances exploration and exploitation by selecting arms based on their upper confidence bounds, taking into account both the estimated rewards and the uncertainty associated with those estimates. The UCB algorithm aims to efficiently identify and exploit arms with potentially high rewards while actively exploring arms to reduce uncertainty. Algorithm 8 shows the working of the UCB algorithm.

UCB

Algorithm 8: UCB()

Input: None

Output: None (just update $w(t)$)

- 1: Initialize/Load $w[|A|]$ and $N[A]$
 - 2: Initialize exploration factor c
 - 3: **for** $i = 1$ to T **do**
 - 4: Sample an action a_t by $\text{argmax}_{a \in |A|}(w(a) + c\sqrt{\frac{\log T}{N(a)}})$
 - 5: Observe reward r_t for action a_t
 - 6: Update $w(a_t)$ based on r_t and increment $N(a_t)$
 - 7: **end for**
-

We start by initializing the estimated reward values for each arm. Also, initialize a count variable for each arm, representing the number of times the arm has been selected as well as the exploration factor c . At each time step, calculate the upper confidence bound for each arm based on its estimated reward value and the confidence interval. The UCB formula is shown in Step 4. The exploration factor c determines the balance between exploration and exploitation. It controls the extent of exploration, with higher values encouraging more

exploration. The square root term in the formula captures the uncertainty associated with the estimated reward value. The $N(a_t)$ term prevents overemphasis on arms that have been selected frequently. We select the arm with the highest UCB value as the chosen action a_t . This balances the exploitation of potentially high-reward arms with the exploration of less-explored arms. After selecting an arm and performing the associated action a_t , we receive a reward r_t . Then, we update the estimated reward for the chosen arm $w(a_t)$ based on the observed reward r_t by using techniques such as sample averaging or incremental updates; and increment the count variable N for the arm a_t .

As the algorithm progresses, the estimated reward values and the counts for each arm are continuously updated. This information is used to calculate the UCB values, which guide the balance between exploration and exploitation. Arms with higher estimated rewards and arms with higher uncertainty due to fewer selections are prioritized. We repeat the algorithm for T iterations. The UCB algorithm optimally trades off exploration and exploitation as it adapts to the observed rewards and uncertainties. As more iterations occur, the algorithm narrows down the range of plausible reward values for each arm and focuses on exploiting the most promising arms.

The UCB algorithm has been widely studied and has theoretical guarantees for achieving near-optimal performance in multi-armed bandit problems. It addresses the exploration-exploitation trade-off by incorporating uncertainty in the decision-making process, providing an effective strategy for a range of real-world applications such as online advertising, recommendation systems, and dynamic pricing.

Regret of the UCB Algorithm

Regret in the context of MAB problems is a measure of how much worse the algorithm performs compared to the optimal strategy. Specifically, it quantifies the difference between the total reward that could have been achieved by always choosing the best arm (the optimal arm) and the reward actually obtained by the algorithm. Below, we provide the regret bounds of the UCB algorithm.

Let's assume we have a multi-armed bandit problem with K arms, and each arm i has an unknown expected reward μ_i . The goal of the algorithm is to maximize the cumulative reward by choosing arms based on observed feedback, and the regret is the difference between the reward obtained by the optimal arm (with the highest expected reward) and the reward obtained by the algorithm.

Define the optimal arm as the arm with the highest expected reward, i.e.,

$$\mu_* = \max(\mu_1, \mu_2, \dots, \mu_K)$$

The regret of the UCB algorithm after T time steps is defined as:

$$R(T) = T\mu_* - \sum_{t=1}^T r_{a_t} \quad (3.6)$$

where, r_{a_t} is the reward received from the arm a_t chosen at time t . The regret bound for the UCB algorithm is given by the following theorem,

Theorem 3.1. [61] *The expected regret incurred by UCB after T rounds is at most*

$$R(T) = \left[\sum_{i:\mu_i < \mu_*} \frac{\ln T}{\Delta_i} \right] + \left(1 + \frac{\pi^2}{3}\right) \left(\sum_{j=1}^n \Delta_j \right)$$

where μ_i is the expected reward of arm i , μ_* is the expected reward of the optimal arm, and $\Delta_i = \mu_* - \mu_i$ is the gap between the expected reward of the optimal arm and arm i .

3.6.2.3 The EXP-3 Algorithm

The **EXP-3 (Exponential-weight algorithm for Exploration and Exploitation)** algorithm is a classic approach used to solve the exploration-exploitation trade-off in the multi-armed bandit setting [61]. EXP-3 is especially preferred when dealing with adversarial or non-stationary environments where other algorithms like UCB or Thompson

Sampling might struggle. These algorithms perform well in stochastic environments but often fail when the reward distributions are actively manipulated or change over time.

In contrast, EXP-3 is robust and adaptable, and it provides theoretical performance guarantees in complex and uncertain settings. It is particularly useful in real-time applications like online advertising, financial trading, or recommender systems, where the environment can change unpredictably and adversarial actors may exist. EXP-3 is designed to handle scenarios where the rewards are adversarial, meaning that an adversary can strategically manipulate the rewards to hinder the learning algorithm's performance. EXP-3 allows for effective exploration while still aiming to maximize the cumulative reward. Algorithm 9 shows the working of the EXP-3 algorithm.

It maintains a list of weights for each action in the action space. Using these weights, it stochastically decides which action to take next, and based on the reward received, it increases or decreases the relevant weights. The values of the actions denote the goodness of that action. We introduce an egalitarianism factor $\gamma \in [0, 1]$, tuning the desire to pick an action randomly. That is, if $\gamma = 1$, the weights do not affect the choices at any step.

EXP-3

Algorithm 9: EXP_3()

Input: None

Output: None (just update $w(t)$)

- 1: Initialize/Load $w[|A|]$
 - 2: $prob_i(t) = (1 - \gamma) \frac{w_i(t)}{\sum_{a=1}^{|A|} w_a(t)} + \frac{\gamma}{|A|}, \forall i \in \{1, 2, \dots, |A|\}$
 - 3: Sample next action act_t stochastically from $[prob_1(t), prob_2(t), \dots, prob_{|A|}(t)]$
 - 4: Observe reward $r_{act_t}(t) \in [0, 1]$ for taking action act_t at t
 - 5: $\hat{r}_a(t) = r_a(t)/prob_a(t)$, if $a = act_t$
 $\hat{r}_a(t) = 0$, otherwise
 - 6: $w_{act_t}(t+1) = w_{act_t}(t) * e^{\gamma * \hat{r}_{act_t}(t)/|A|}$
-

As per the Algorithm 9, If the w is empty (at the start of the training), then initialize it with suitable values; otherwise, load the previously created w into memory. As described earlier, the dimension of this list is $|A|$ (the size of action space A). Next, we weigh the actions based on the corresponding values stored in the w . The probability of selecting an action i at time t ($prob_i(t)$) is directly proportional to the corresponding action weight at time ($w_i(t)$). Here, an egalitarianism factor $\gamma \in [0, 1]$ also plays a role in action selection; $\gamma = 0$ would calculate probabilities purely based on *table* values, while $\gamma = 1$ would assign the same probability to each of the actions. After calculating the probabilities for each action i , in step 3, the algorithm stochastically picks one action based on the calculated probabilities. In step 4, the algorithm observes the reward $r_{act_t}(t)$ for taking action act_t at time t . After that, in step 5, the algorithm updates the reward based on whether the action was selected or not; the new reward function $\hat{r}_{a_t}(t)$ is inversely proportional to the probability $prob_{a_t}(t)$. If the action was not selected, then the $\hat{r}_{a_t}(t)$ is set to zero, as expected. Finally, in step 6, the algorithm updates the w ; only the state-action pair that got selected at time t gets updated, while other values in w remain unchanged. These updates are exponential in nature and proportional to the new reward $\hat{r}_{a_t}(t)$.

The EXP-3 algorithm deals with the explore-exploit dilemma by stochastically selecting an action based on the calculated probabilities in Step 3. This step ensures that the best-known action is picked with higher probability while also occasionally selecting 'not so good' actions. After selecting any action and getting the corresponding reward in that state, the reward is weighed with respect to the probability. A reward for low-probability actions gets enhanced even further, allowing the algorithm to revisit those actions. Thus, the EXP-3 algorithm visits all the state-action pairs a sufficient number of times.

3.7 Summary

In this chapter, we presented a detailed discussion of reinforcement learning. We started by describing the reinforcement learning framework with the help of a few examples. Then, we stated some of the important definitions of RL. We broadly focused on two types of RL methods: tabular RL and approximate solution RL methods. In tabular RL methods, we detailed three classes of RL techniques, namely, Dynamic Programming, Monte Carlo, and Temporal-Difference Learning. In approximate solution methods, we presented policy gradient-based RL methods; more precisely, we explained the DDPG method. Finally, we provided a detailed overview of MAB-based techniques along with a description of important MAB algorithms. With the foundational understanding of game theory and reinforcement learning established in the previous chapters, we now shift focus to the smart grid domain. In the next chapter, we provide an in-depth discussion of the smart grid, accompanied by a comprehensive review of relevant literature to contextualize our technical contributions.

Chapter 4

Smart Grids: An Overview, PowerTAC Simulation, and Literature Review

Smart grid technology is an emerging technology aimed at rectifying the drawbacks of conventional grid systems. A smart grid system promotes active participation from the consumers in the grid's operations, unlike conventional grids, where consumers only consume electricity. Smart grid technology promotes wide-scale automation in different parts of the grid's operations, and designing strategies to automate smart grid operations is an active research area. However, to validate the efficacy of any novel strategy in the real-world smart grid system is impractical due to the system's complexity. Any fault in the strategy could disrupt the grid system and impact thousands of consumers. To remedy this and promote smart grid research, **Power Trading Agent Competition (PowerTAC)** designed an efficient and close-to-real-world simulator that incorporates all the crucial elements of the smart grid system. In this chapter, we discuss smart grids

and PowerTAC simulators in detail. Finally, we present a literature survey of the strategies proposed for smart grid systems.

4.1 Introduction to a Smart Grid System

The advent of technology has revolutionized the way we interact with our surroundings, and the energy sector is no exception. One such remarkable advancement is the smart grid system, a modernized approach to managing and distributing electricity. Unlike the conventional grid system, which primarily focuses on one-way power flow from centralized power plants to consumers, the smart grid integrates advanced sensing, communication, and control technologies to create a *two-way information flow* between consumers and utility providers. This innovative system enables greater efficiency, reliability, and sustainability in the generation, distribution, and consumption of electricity while empowering consumers to actively participate in energy management and make informed decisions about their usage. By harnessing the power of *digitalization*, the smart grid system represents a significant leap forward in modernizing our electrical infrastructure and addressing the challenges of the 21st century. Formally, smart grid technology is defined as,

Definition 4.1.1: Smart Grid [11]

A smart grid is an electricity network based on digital technology that is used to supply electricity to consumers via two-way digital communication, which allows for monitoring, analysis, control and communication within the supply chain to help improve efficiency, reduce energy consumption and cost, and maximize the transparency and reliability of the energy supply chain.

Smart grid systems aim to overcome the weaknesses of conventional grids by enabling customers to get involved in using smart meters. The smart grid system encompasses

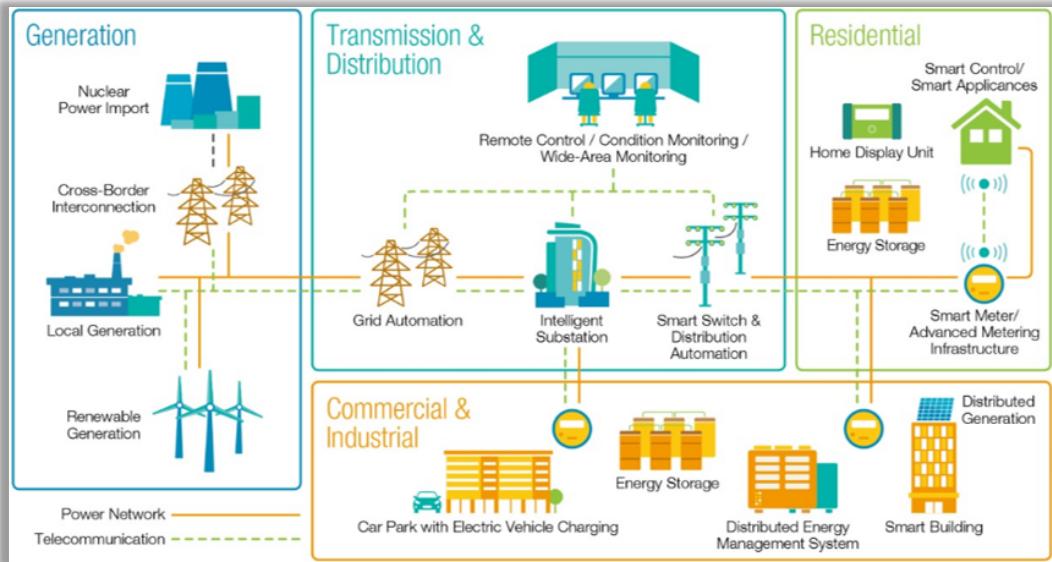


Figure 4.1: Overview of Smart Grid Technology [Image Credit: © [5]]

wholesale and **retail markets** along with **distribution utility** and **electricity brokers**, each playing a vital role in the efficient and dynamic management of electricity. These markets facilitate the buying and selling of electricity, ensuring a reliable supply of power to consumers while optimizing grid operations and promoting the integration of renewable energy sources.

The wholesale market of a smart grid system primarily focuses on the bulk purchase and sale of electricity between power generators, electricity brokers, and retail consumers. This market operates at a regional or national level and relies on complex mechanisms like **double auctions** to establish prices and manage the supply-demand balance. Wholesale electricity markets leverage advanced technologies and sophisticated algorithms to enable efficient market clearing, where electricity generation is matched with demand in real-time or through forward contracts. The wholesale market also plays a crucial role in enabling the integration of renewable energy sources into the grid. A generation from renewable resources like solar and wind can be intermittent and variable; the wholesale

market provides a platform for renewable energy producers to sell their excess energy to other market participants when their generation exceeds their local demand.

On the other hand, the retail market of a smart grid system focuses on the distribution and sale of electricity to end consumers, such as residential, commercial, and small industrial customers. It involves the interaction between the broker and consumers, enabling the billing, metering, and provision of electricity-related services. The retail market allows consumers to choose their *electricity provider (broker)*, offering them more control and flexibility over their electricity consumption. In the retail market, *smart meters* enable accurate measurement of electricity consumption and real-time monitoring. This information helps consumers understand their electricity usage patterns, identify areas for improvement, and make informed decisions to optimize their consumption and reduce costs. Additionally, the retail market facilitates the implementation of *demand response* programs, where consumers can voluntarily adjust their electricity usage during peak periods in exchange for incentives or lower rates, thus contributing to load balancing and grid stability. The retail market of a smart grid system also encourages the adoption of distributed energy resources such as rooftop solar panels and home energy storage systems. Consumers can generate their own electricity, store excess energy, and even sell it back to the grid, fostering a more decentralized and sustainable energy ecosystem.

The distribution utility is responsible for electricity distribution from the wholesale market to customers in the retail market. Moreover, it is in charge of handling the *supply-demand imbalance* scenarios in the market. The *electricity brokers* play a significant role in facilitating the efficient and transparent buying and selling of electricity between market participants. In smart grids, electricity brokers are the electricity distribution companies that procure electricity by bidding in the wholesale market auction and selling the procured electricity via tariff contracts. These brokers act as intermediaries, connecting electricity buyers and sellers and enabling wholesale and retail transactions. They leverage

advanced technologies and market knowledge to optimize the procurement and pricing of electricity, promoting grid stability, cost-effectiveness, and sustainability.

We need an effective simulator to test any novel strategy for the smart grid system, as testing the strategies directly on real-world smart grids is physically impractical. To help conduct research on smart grids, Ketter et al. designed an efficient smart grid simulator called Power Trading Agent Competition (PowerTAC) [6, 62, 63, 64]. Below we describe the PowerTAC simulator in detail.

4.2 The PowerTAC Simulator

PowerTAC models the high complexity of contemporary and future retail electricity markets, allowing for large-scale experimentation. The system diagram of PowerTAC is shown in Figure 4.2, which includes all the crucial elements of a real-world smart grid system. As shown in the figure, PowerTAC models a *wholesale market, retail market (or tariff market), balancing market, distribution utility* and *electricity brokers*. The PowerTAC wholesale market consists of large energy suppliers (power generating companies or GenCos), and the retail market incorporates various types of consumers and producers. The distribution utility owns and operates the local grid and transfers the electricity from wholesale to the retail market and within the retail market. PowerTAC balancing market takes care of the real-time balancing of electricity supply and demand. Note that PowerTAC introduces the balancing market for the separation of responsibilities. Such a market does not exist in real-world smart grids; the distribution utility performs the balancing operations. The electricity brokers are the distribution companies and play a crucial part in the PowerTAC simulation. They interact with all three PowerTAC markets and are instrumental in making the PowerTAC smart grid simulation function. Below we describe the key components of the PowerTAC simulator in detail.

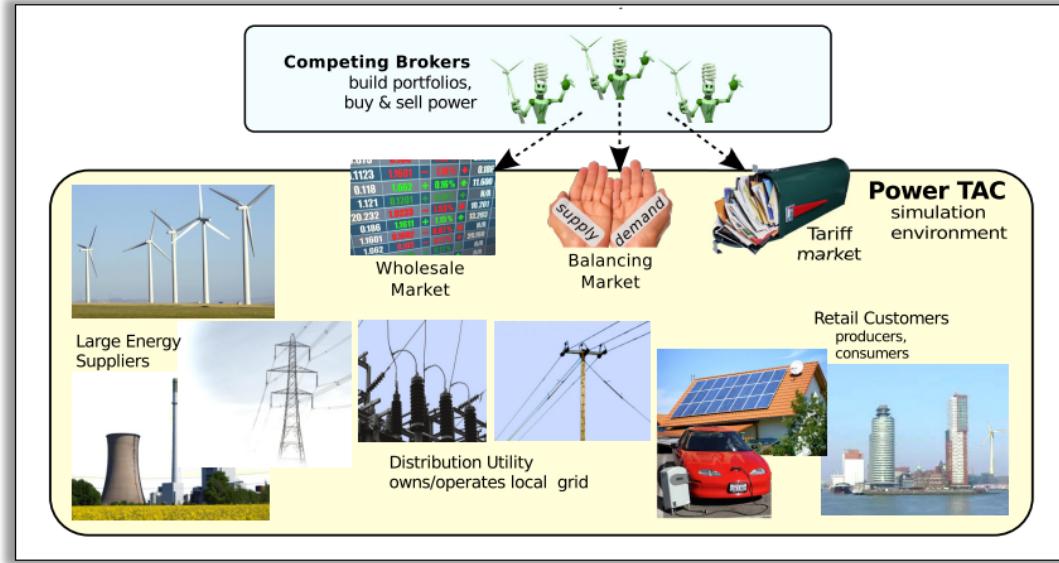


Figure 4.2: Major Elements of the PowerTAC Scenario [Image Credit: © [6]]

4.2.1 Information from the Simulator

Brokers in the PowerTAC simulation get feedback on their actions via messages from the server. Some messages are public and sent to all participating brokers, whereas some are private and sent to individual brokers. The following information is made public at the start of each game.

- ***Game Parameters:*** The parameters used to configure the current game.
- ***Broker Identities:*** The identities of all competing brokers in that particular game.
- ***Customers List:*** Names and properties of customers in the current game, most importantly, their power types.
- ***Default Tariffs:*** Tariffs offered by the default broker, to which customers are subscribed at the beginning of the game and can re-subscribe at any point in the game. There are two such tariffs, one for producers and one for consumers.

- **Bootstrap Customer Data:** Consumption and production of each customer during the bootstrap phase under the default tariffs.
- **Bootstrap Wholesale Market Data:** Total cleared quantity and clearing price of the wholesale market during the bootstrap period. This is a result of default broker bidding in the wholesale market.
- **Bootstrap Weather Data:** Weather reports for the entire bootstrap phase.

The following information is communicated publicly every 6 timeslot:

- **Tariff Updates:** New tariffs, revoked tariffs, and superseding tariffs submitted by all brokers.

The following information is communicated privately every 6 timeslots:

- **Tariff and Subscription Changes Transactions:** Tariff publication/revocation fees, customer subscription changes (signup or withdraw), and associated charges/bonuses (early-exit penalty or signup bonus).

The following information is communicated publicly every timeslot.

- **Wholesale Market Clearing Data:** Clearing prices and total cleared quantities for each of the 24 PDAs in the wholesale market.
- **Wholesale Market Orderbooks:** Orderbooks (uncleared bids and asks) for each of the 24 PDAs in the wholesale market.
- **Total Electricity Production and Consumption:** Total electricity production and consumption for the current timeslot.
- **Weather report and Weather Forecast:** Weather conditions for the current time slot and forecast for the next 24 timeslots.

The following information is communicated privately every timeslot:

- **Balancing Transactions:** Broker imbalance amount and associated charge/credit from the balancing market.
- **Portfolio Usage and Payment Transactions:** Subscribed customer usage records for the past timeslot and associated charges/payments according to subscribed tariffs.
- **Distribution Transactions:** Broker's electricity distribution quantity among its subscribed customers and associated charges levied by DU.
- **Wholesale Market Transactions:** Cleared or partially-cleared bids and asks submitted by the broker.
- **Wholesale Market Positions:** Electricity commitments made by the broker in the wholesale market which specify the energy to be delivered by/to the broker in future timeslots.
- **Cash position:** Broker's current bank balance.

The following information is communicated publicly every 168 timeslots (1 week):

- **Threshold Demand and Peak Timeslots with Demand:** The threshold demand for capacity charge assessment and the top 3 peak net demands in the market along with their respective timeslots.

The following information is communicated privately every 168 timeslots (1 week):

- **Capacity Charges:** Transmission capacity fees and the associated amount by which the broker exceeded the threshold demand in the 3 peak demand timeslots (if any).

4.2.2 The PowerTAC Wholesale Market

PowerTAC's wholesale market contains large power-generating companies, also known as GenCos. These GenCos produce electricity in bulk and sell it at wholesale prices. The electricity in the wholesale market is sold through an auction mechanism. An auction is a process of buying or selling goods or items; here, it is an electricity auction. More precisely, PowerTAC organizes a **day-ahead periodic double auction** for the electricity trade between GenCos and electricity brokers. Here **day-ahead** indicates that the brokers buy or sell electricity for the future delivery timeslots, typically between 1 to 24 hours in the future. These auctions are periodic, meaning auction-clearing happens periodically after every simulated hour. The double auction requires two players to participate in the auction, buyer and seller. PowerTAC uses a particular type of double auction called ***k*-double auction**, which is explained below. The GenCos simulate the demand of a regional market that is significantly larger than the demand generated by the PowerTAC simulation scenario.

4.2.2.1 *k*-Double Auction

k-double auction is a type of auction for buying and selling resources, where potential buyers submit their bids, and potential sellers submit their asks to the auctioneer. The auctioneer receives all the bids and asks and determines each player's **clearing quantity** and the **clearing price** using a specific allocation rule and payment rule. Below are some important definitions to understand *k*-double auction.

Definition 4.2.1: Clearing Price

A **clearing price** is a price at which the auctioneer clears the market after matching the potential bids of the buyers with the potential asks of the seller.

Definition 4.2.2: Clearing Quantity

A **clearing quantity** of a player is the number of items a buyer[seller] receives[sells] from the auction after clearing.

Definition 4.2.3: Allocation Rule

An **allocation rule** determines the quantity bought by each buyer and the quantity sold by each seller in an auction. Basically, the allocation rule determines the clearing quantity of each player.

Definition 4.2.4: Payment Rule

A **payment rule** determines the payment each player has to make (buyer pays and seller earns) at the time of auction clearing.

Definition 4.2.5: k -Double Auction

If a buyer and seller participate in a double auction, and if the buyer's bid b is higher than the seller's bid s , then the clearing price is given by $kb + (1 - k)s$ for some fixed $k \in [0, 1]$.

Figure 4.3 shows an example of market clearing in the PowerTAC wholesale market using the k -double auction. During each auction instance, buyers and sellers submit their bids and asks to the auction mechanism. As per the PowerTAC convention, bids are submitted with a positive electricity amount and negative price, whereas asks have a negative electricity amount and positive price. Not all bids/asks require a specific price; the bids/asks without price details are known as market orders and are sorted first (as they are treated as the highest bid amount or lowest ask amount).

Demand and supply curves are constructed from the submitted bids and asks for that particular auction instance. The asks are sorted in the increasing order of the prices (supply curve), and bids are sorted in the decreasing order of the price (demand curve), as shown

in the figure. The clearing price is at the intersection of the demand and supply curves. If there is no unique price where the supply and demand curves intersect, as in this example, then the clearing price is calculated based on the lowest usable bid (b) and the highest usable ask (s) price by following the definition of *k -double auction*. In PowerTAC, the value of k is set to be 0.5. Thus, the clearing price is the mean value of b and s . In the example, the lowest usable bid is **bid 8** and the highest usable ask is **ask 6**; thus, the clearing price is set to be the mean of bid 8 and ask 6 (shown with a red dot in the figure). So, in the example, asks 1-6 are all matched by higher-priced bids, and bids 1-8 are all matched by lower-priced asks. However, out of 5 MWh of ask 6, only the first 2 MWh is matched. The remaining ask 7, and bids 9-10 cannot be matched. After clearing the auction this way, the **clearing price** is 16 (mean of $b = 17$ and $s = 15$), and **total cleared quantity** is 27 MWh.

PowerTAC follows the **uniform payment rule** in which all the successful buyers pay the clearing price, and all the successful sellers receive the clearing price irrespective of their bids/asks. The cleared quantity for a buyer/seller is the same as its bid/ask electricity amount in case of full clearing and partial amount in case of partial clearance of bid/ask. In the example, all the successful bids (bids 1-8) pay the clearing price of 16, and all the bids are fully cleared. The successful asks (asks 1-6) receive the price 16, and ask 1 to 5 are fully cleared, whereas ask 6 is partially cleared (2 MWh out of 5 MWh).

4.2.3 The PowerTAC Tariff Market

PowerTAC's tariff or retail market comprises various types of customers, such as consumers, producers, prosumers, storage, electric vehicles, etc. The consumers vary from households, offices and villages to hospitals. Some consumers have the ability to curtail their usage given incentives, called interruptible consumers. Producers are modelled to generate electricity from renewable resources like wind turbines and solar panels. Prosumers are the customers who can both produce electricity as well consume it, i.e., a house

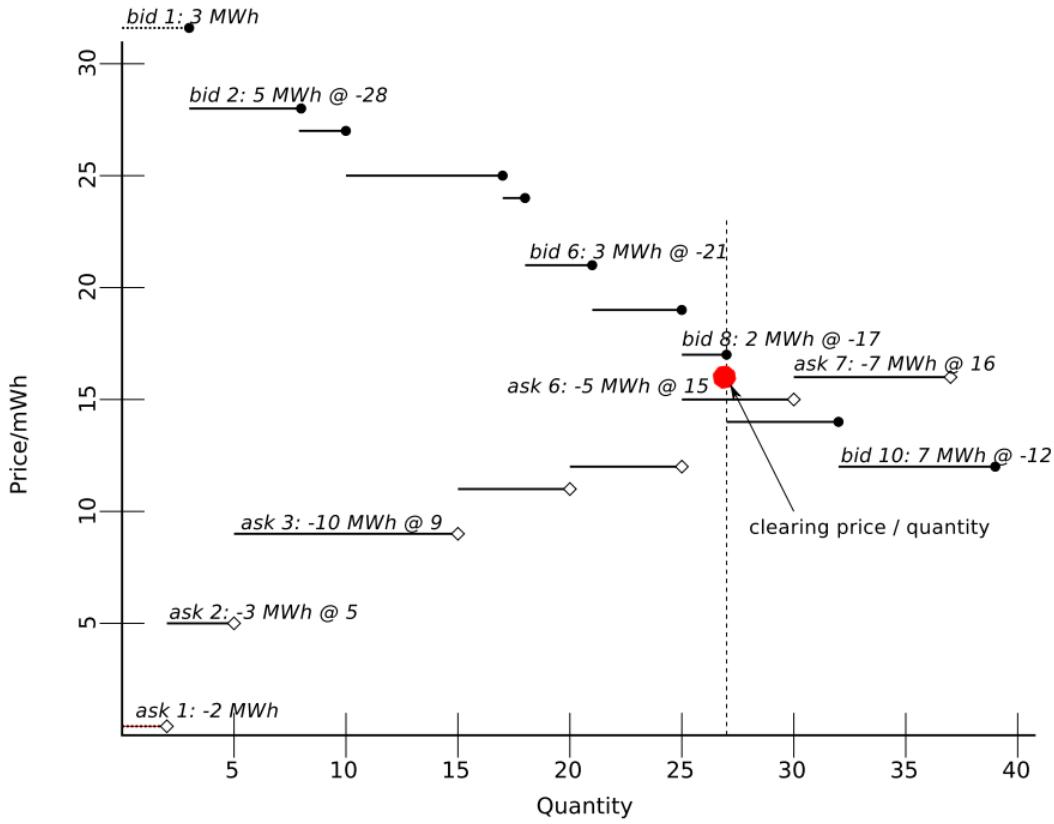


Figure 4.3: Market clearing example: bid 8 and part of ask 6 are the last to clear [Image Credit: © [6]]

with a solar rooftop. Storage customers use batteries or electric vehicles to store electricity that can be supplied to the grid whenever needed. These customers are state-of-the-art customer models and behave similarly to real-world customers, thus, effectively mimicking the real-world smart grid's tariff market.

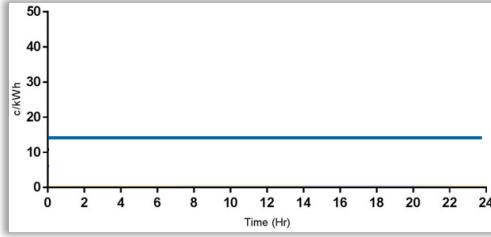
These customers subscribe to one of the brokers in the tariff market by accepting the tariff contract published by that broker. Brokers offer various types of tariffs to attract different types of customers. Each customer can subscribe to the tariff offered for its type; when multiple tariffs are available in the market, the customer can evaluate the tariff

and join the most suitable tariff based on its usage pattern. These customers can keep evaluating tariffs periodically and switch to the most attractive tariff. While publishing a tariff, a broker needs to specify the customer type for whom the tariff is generated. The other types of customers can not subscribe to that tariff.

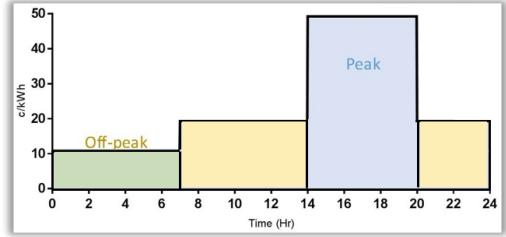
There are multiple tariff types available for a broker, namely, ***Fixed Price Tariff (FPT)*** (Figure 4.4a), ***Time-of-Use (ToU) tariff*** (Figure 4.4b), ***Tier tariff*** (Figure 4.4c), and ***Variable tariff*** (Figure 4.4d). Fixed price tariffs are the tariffs in which customers pay a fixed price throughout the tariff usage period. In the Time-of-Use tariff, customers have to pay the price depending on the hour of use. ToU tariffs can be a daily ToU or a weekly ToU. In Tier tariffs, customers pay a certain price up to a usage threshold; after that, they pay a different price once that usage threshold is crossed. There can be multiple such thresholds in the tariff contract. Variable tariffs allow brokers to dynamically decide the price a few hours ahead of the usage interval. A broker can offer a combination tariff involving two or more types of tariffs as well.

4.2.4 The PowerTAC Balancing Market

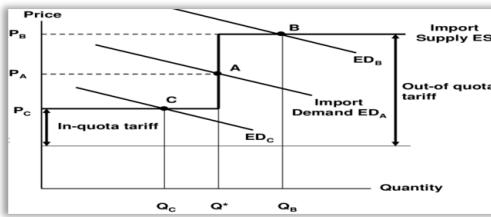
PowerTAC's balancing market is responsible for the real-time balancing of the supply and demand of electricity in the PowerTAC simulation. This market penalizes the broker heavily for any imbalance of supply and demand in its portfolio; this way, the market incentivizes the brokers to balance their own portfolios of electricity supply-demand in each timeslot. There are two ways for brokers to exploit the balancing market: (i) by having a net imbalance that decreases the overall imbalance and (ii) by offering storage and flexible load resources from customers who have agreed to external controls given some incentives.



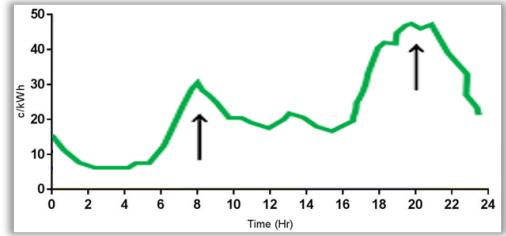
(a) Fixed Price Tariff



(b) Time of Use Tariff



(c) Tier Tariff



(d) Variable Tariff

Figure 4.4: Types of Tariffs

4.2.5 The PowerTAC Broker

The brokers are the crucial element in the PowerTAC simulator. A broker interacts with all three markets of the simulator. Precisely, it procures electricity from the wholesale market, sells electricity in the retail market, and, while doing so, maintains the supply-demand balance in the balancing market. The brokers' activities are described in detail in Section 4.3.

4.2.6 The PowerTAC Competition

A crucial feature of PowerTAC is that the simulator provides a way for several autonomous electricity brokers to compete to make profits by operating in tariff, wholesale, and balancing markets. An *annual PowerTAC tournament* is organized wherein multiple teams deploy autonomous brokers competing in all three markets. The tournament

consists of several games organized between brokers in different player configurations and varying weather conditions. The duration of a game is around ***60 simulation days***, and the simulation time is discretized into time slots corresponding to every hour of the day. During the game, a broker agent aims to develop a subscriber base in the retail market by offering competitive tariffs, which could be fixed price (FPT), tiered, time-of-use (ToU), variable or a combination of various types of tariffs. Brokers satisfy the electricity requirement of their subscriber base by buying power in the wholesale market through day-ahead PDAs. A broker can participate in ***24 double auctions*** corresponding to consumption slots that could be one to twenty-four hours away at any simulation time. Through subscriptions to storage customers, brokers can also manage grid imbalances.

An important thing to note about PowerTAC is ***Capacity Transactions***. In PowerTAC, capacity transactions are the penalties incurred by the brokers if the customers subscribe to their portfolio and contribute to the peak demand scenarios. These huge penalties are a way to penalize the brokers for letting the customers create demand peaks. The demand peaks are undesirable in the smart grids, as during the demand peaks, the GenCos need to ramp up their electricity production, which is a costly operation; thus, brokers get penalized in proportion to their contribution to demand peaks.

The broker with the ***highest normalized score*** is declared the ***tournament winner***. The normalized score is the summation of scores a broker achieves in all different player configurations in the tournament. The scores are calculated based on a broker's cash position (bank balance) at the end of each game. Thus, each broker aims to ***maximize its profits*** during the games to maximize the cash position. To maximize the profit, a broker needs to maximize revenue in the tariff market while minimizing its electricity procurement cost in the wholesale market, along with balancing penalties and capacity transaction penalties.

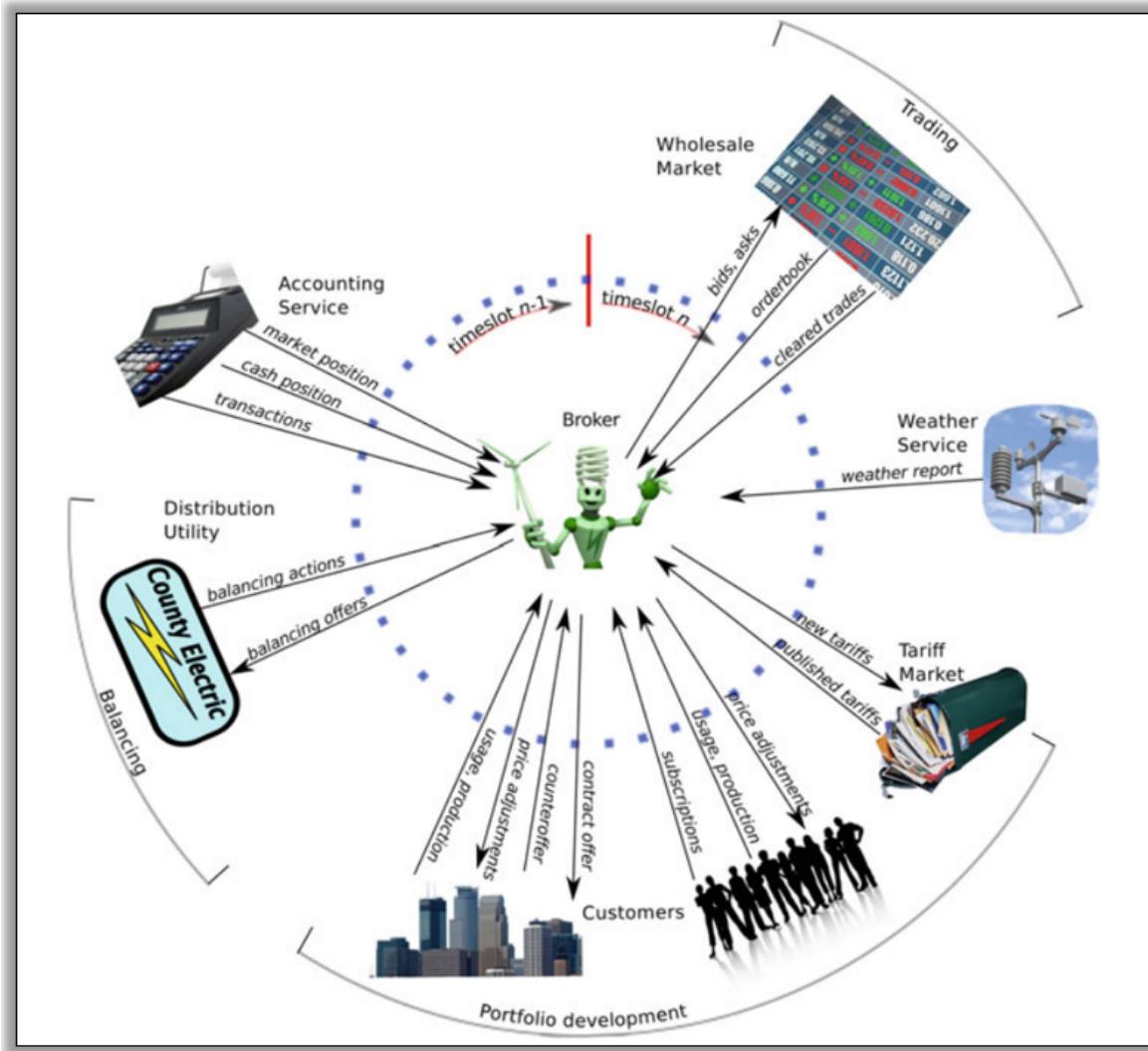


Figure 4.5: Broker's Activities in PowerTAC Markets [Image Credit: © [6]]

4.3 A Typical Broker's Activities in the PowerTAC

As stated earlier, an electricity broker or broker plays an important role in a smart grid system and is essential for the smooth functioning of the PowerTAC simulator. A broker carries out various activities in each of the three PowerTAC markets; Figure 4.5 shows the activities a broker needs to perform during the PowerTAC game.

4.3.1 A Broker in the Wholesale Market

The main task of a broker in the wholesale market is to design a *bidding strategy* to participate in the day-ahead PDA to minimize electricity procurement costs. As the auctions are day-ahead auctions, a broker needs to participate in a series of auctions, and therefore, a bidding strategy involves planning across current and future auctions. On top of it, the auctions are *real-time*, and a broker has limited time to place the bids; thus, the strategy should be less compute-intensive to make real-time decisions. For example, in PowerTAC, a broker gets a window of 5 seconds, within which it needs to place bids along with its other operations in tariff and balancing markets. Any bidding strategy is concerned with deciding two parameters, *bid price* and *bid quantity*, to place a bid in the auction. As it is a day-ahead auction, a broker has 24 opportunities to procure the required quantity. For example, an electricity broker participating in day-ahead markets can procure electricity for a delivery timeslot at any prior time within 24 hours (at a period of every one hour). The leftover quantity that a broker could not procure must be bought from the balancing market at much higher prices; similarly, any extra quantity procured by a broker may be sold in the balancing market at a nominal price. Thus, a broker needs to carefully design its forecasting algorithms to accurately predict customer demands and bidding strategy to procure the forecasted demands from the wholesale auction as much as possible.

4.3.2 A Broker in the Tariff Market

The major activity for a broker in the tariff market is to publish *lucrative tariff contracts* to build the portfolio of customers. The tariff contracts should cover the electricity procurement costs with some profit margins and be attractive to customers. In PowerTAC, brokers and customers can see all the active tariffs in the market; thus, a broker needs to periodically update its tariffs to remain competitive in the market. A broker can either revoke its previously active tariff or publish a new one along with keeping the previous one

active in the market. A broker needs to store the accounting information of all its active tariffs to check the demand and profitability. Low-subscribed or loss-making tariffs should be removed, and more suitable tariffs should be published. In the tariff market, a broker should aim to create a *diverse portfolio of various customer types*, as a balanced portfolio helps in mitigating balancing and capacity transaction penalties.

4.3.3 A Broker in the Balancing Market

Brokers may submit balancing orders to the balancing market in order to trade electricity. A broker can benefit from the balancing market if it ends up on the opposite of the market imbalance. For example, if the market is in a deficit of electricity and the broker has additional electricity stored in the storage devices, then it can place a *balancing order* to sell electricity to the market and vice versa.

4.4 Liturature Survey

In this section, we present a literature survey of the strategies used in double auctions, PowerTAC's wholesale and tariff markets, as well as strategies used for peak reduction. Here, we provide a detailed explanation of some of the most relevant works while giving a brief overview of other related works.

4.4.1 Bidding Strategies in Double Auctions

Here, we summarise a few important papers in Double Auction literature that can be useful in modelling novel strategies for the smart grid wholesale market's PDAs.

Bilateral Trade with the Sealed Bid k -Double Auction [25]

k -double auction determines the terms of trade, where k is in the unit interval. The buyer and seller submit bids b and s , respectively. If b exceeds s , trade occurs and at price $kb + (1 - k)s$. This trade scenario is modelled as a *Bayesian Game*, where each trader

privately knows his valuation and only has some belief about the opponent’s valuation. This paper addresses the problem of finding the existence of a multiplicity of equilibria for the k -double auction for a generic class of traders’ beliefs (it particularly looks for the differential equilibrium strategies). It also examines the efficiency of the proven equilibrium for generic beliefs.

The authors aim to provide a differential equilibrium for a k -double auction. For that, they start by defining important concepts and constraints, such as regular equilibrium, restriction on the distribution of traders’ CDF, constraints on traders’ strategies, allocation, and payment rules. Then using the definition of interim efficiency (because a trader knows his valuation and has some beliefs about the opponent’s valuation) and equilibrium, they come up with differential equations that guarantee equilibrium when traders’ strategies follow those equations. The authors also explain the geometry of the solution for some selected distributions. After proving the existence of equilibrium, they analyze the efficiency of these equilibria at ex-post, interim, and ex-ante stages. They provide results for both $k \in \{0, 1\}$ and $k \in (0, 1)$ cases for each of the categories. In this work, the authors present theorems underlining the existence of equilibria and the set of conditions under which these equilibria hold. They further provide analysis to theoretically showcase the efficiency of these equilibria at ex-ante, interim, and ex-post stages.

This work extends the ideas presented in some of the previous works and also refers to a few important results from the previous papers. Myerson and Satterthwaite [65] showed that the private information and individual incentives of the two traders imply that neither the k -double auction nor any other trading mechanism can be ex-post classical efficient. Chatterjee and Samuelson [14] showed that ex-ante performance can be achieved for uniform distribution of valuations and $k = 0.5$. Theorem 5.2, which is from Myerson [66] and Williams [67], states that the buyer’s bid/seller’s offer auction achieves ex-ante efficient performance for all pairs of distributions for $k \in \{0, 1\}$. Chatterjee and Samuelson [14] first examined Bayesian equilibria within the k -double auction. Myerson and Satterthwaite [65]

applied to double auctions with two-sided uncertainty. Wilson [68] established that as the number of traders grows large, the k -double auction achieves interim efficient performance if well-behaved equilibria exist. Leininger et al. [69] use different techniques to explore the equilibria in the bilateral k -double auction. Remarkably, they show the existence of a large class of equilibria that involve step-function strategies. Vetsikas [70] attempts to find equilibrium strategies for multi-unit sealed bid auction for m^{th} and $(m+1)^{th}$ price sealed bid auction.

The novelty of this work extends the idea of previous papers on finding equilibrium and analyzing efficiency for a general class of distributions. This paper provides proof for the existence of differential equilibrium and attempts to examine the geometry of the solution. This paper provides the conditions for interim and ex-ante efficiency.

Bargaining Under Incomplete Information [14]

This paper is the precursor of the previous paper [25]. Uncertainty about the opponent's walkaway price is frequently encountered in bargaining situations; the better the bargaining information about the opponent, the better a player expects to fare in the negotiations. Probabilistic assessments become more complicated in the environment with stochastic dependence between the player values. Towards this, this paper analyzes the bargaining model of bilateral monopoly under uncertainty. The Nash equilibrium solution is proposed and issues of relative bargaining advantage and efficiency are examined for a number of special cases.

The authors start by proving the result that equilibrium bargaining strategies are increasing in reservation prices. Then they provide the linked differential equations that strategies must satisfy to reach equilibrium strategies. They also provide the necessary part of the theorem, while a sufficient part is presented in [25]. After that, they analyze a few special cases (uniform distribution of valuations) using the above results.

They provide some important results for the conditions that need to be satisfied for the linked differential equations. One of the important theorems says,

Theorem 4.1 (*Chatterjee and Samuelson [14]*). *Under the sealed offer bargaining rule, the equilibrium bargaining strategies of buyer and seller are increasing in the respective reservation prices.*

The novelty of this work lies in a detailed analysis of special cases of equilibrium and a discussion of the limitation of single-stage bargaining and how multiple rounds of bargaining would make agreements more frequent. Unlike the previous paper [25], this paper only focuses on valuation functions following a uniform distribution.

This paper mentions some of the fundamental papers of the game-theoretic literature on bargaining. The previous work has assumed that the participants possess complete information about the negotiation situation; towards this, some of the important contributions are provided by Nash [71], Harsanyi [72], Schelling [73], Cross [74], and Roth [75]. Young [76] provides a summary and critique of the most important literature on the bargaining problem.

4.4.2 In the PowerTAC Wholesale Market

The PowerTAC wholesale market poses an exciting challenge due to the intricacies of real-time day-ahead PDAs. The majority of the strategies proposed in the literature are specifically designed for the continuous double auctions, as opposed to the periodic double auctions being used in the PowerTAC (or, in general, smart grid) wholesale market. Thus, one may need to figure out ways to mould the existing bidding strategies for PDAs or to design entirely novel bidding strategies. Below, we present some of the widely accepted wholesale bidding strategies in PowerTAC, followed by a brief survey of other related bidding strategies.

TacTex'13: A Champion Adaptive Power Trading Agent [77]

TacTex aims to maximize its utility by solving simultaneous optimization problems over a combination of decision variables: (i) electricity selling price, (ii) total electricity de-

mand of its customers, and (iii) electricity procurement cost. Solving this optimization problem is intractable, therefore TacTax approximates a solution by solving two independent optimization problems as follows, (i) Optimize cost given predicted demand and electricity-selling price, and (ii) Optimize demand and selling price given predicted cost. These two optimization problems are still intractable, and they are being approximated by TacTex's Wholesale Market Module (approximates problem (i)) and Tariff Market Module (approximates problem (ii)). TacTex approximates the above two optimization problems individually. The wholesale market module of TacTex is presented here, while the tariff market module is diverted to Section 4.4.3.

In the wholesale market module, TacTex models the problem in the MDP framework and solves using dynamic programming. This method allows for high reuse of data, and thus quick online learning with little data. Importantly, this method results in an online Reinforcement Learning bidding algorithm, enabling TacTex to adapt and optimize its bidding strategy based on each game-specific market condition.

In the wholesale market, TacTex optimizes the costs of procuring the electricity needed to satisfy the predicted demand resulting from selling prices at any given timeslot. TacTex models the sequential bidding process in the MDP framework and solves this MDP using the Dynamic Programming technique, as shown below.

TacTex considers the following MDP in the wholesale market:

- ***State:*** $s \in \{0, 1, \dots, 24, \text{success}\}$, $s_0 = 24$
- ***Actions:*** $\text{limit - price} \in \mathbb{R}$
- ***Transition:*** a state $s \in \{1, \dots, 24\}$ transitions to one of two states;
 - if a bid is partially or fully cleared, transition to *success*
 - otherwise, transition from state s to state $s - 1$
- ***Reward:*** Here, both balancing-price and limit-price are taken as negative, so maximizing the reward results in minimizing costs.

- If state=0, the reward is the per-unit balancing price
 - If state=*success*, the reward is the limit price of the successful bid
 - Otherwise, In states $s \in \{1, \dots, 24\}$, the reward is 0
- **Terminal States:** $\{0, success\}$

Since the MDP is linear, solving it requires one back-sweep, starting from state 0 back to state 24, applying the following backup operator to compute a value function:

$$V(s) = \begin{cases} \text{balancing - price} & , \text{if } s = 0 \\ \min_{lp} \{ p_{cleared} \times lp + (1 - p_{cleared}) \times V(s - 1) \} & , \text{if } s \in \{1, \dots, 24\} \end{cases}$$

Where, $p_{cleared}$ is calculated using the following formula:

$$p_{cleared}(s, lp) = \frac{\sum_{tr \in trades[s], tr.cp < lp} tr.cleared_electricity}{\sum_{tr \in trades[s]} tr.cleared_electricity} \quad (4.1)$$

Thus, $p_{cleared}(s, lp)$ is a ratio of cleared electricity when *clearing-price* (*cp*) of the trade is less than limit-price (*lp*) for the trades in state s and totally cleared electricity in all the trades in state s . The MDP's solution determines an optimal limit price (*lp*) for each of the 24 states, and the electricity amount is set to the difference between predicted demand and the electricity that is already procured for each of the 24 states.

TacTex's wholesale strategy is a modified version of Tesauro's bidding algorithm [78]. But unlike Tesauro's bidding algorithm, in TacTex's MDP model, energy amount is not a part of the decision-making; TacTex always places a bid to buy energy amount equal to the difference between predicted demand and the energy that is already procured for the target timeslot. Secondly, TacTex makes a transition to a terminal state even when the bid is partially cleared. This enables all the instances of the sequential bidding process to execute over the same sequence of states, which allows for computational and data efficiency.

An Intelligent Broker Agent for Energy Trading: An MDP Approach [79]

This paper presents the idea and architecture of PowerTAC 2012 broker AstonTAC. Aston-TAC employs an MDP-based strategy in the wholesale market with the goal of minimizing

energy purchase costs. For that, the clearing price needs to be predicted in order to buy the energy when the clearing price is low. AstonTAC uses the ***Non-Homogeneous Hidden Markov Model (NHHMM)*** to predict wholesale market clearing prices. AstonTAC uses techniques inspired by Monte Carlo methods to learn model parameters like rewards and transition probabilities. It uses the pay-off average to decide the optimal action to take instead of computing the transition probabilities and the expected reward values.

AstonTAC is built to adapt to environmental changes and to act autonomously during the simulation. AstonTAC employs MDP to determine bids in the wholesale market and to calculate the volume to buy; it uses customer usage prediction using HMM. AstonTAC has following modules: (1) Clearing Price Predictor (to predict the clearing price for future auctions), (2) Customer Energy Production Prediction (to predict energy production of contracted customers), (3) Customer Energy Consumption Prediction (to predict energy consumption of contracted customers), (4) Customer Energy Demand Manager (to determine net demand for each auction hour), and (5) AstonTAC MDP (to place bids in the wholesale market). Below, we highlight the wholesale module of AstonTAC.

AstonTAC models an MDP in the wholesale market, which suggests a bid (price and quantity) for each timeslot (actually, MDP is only responsible for quantity decisions; price decisions are taken using clearing price predictor). The aim of this MDP is to buy energy at a low price and keep the imbalance as low as possible.

AstonTAC considers the following MDP in the wholesale market:

- ***State:*** The state is represented by the *number of hours ahead* (proximity), *the remaining quantity of energy needed* $\in [0, 1]$ (a ratio of remaining quantity and predicted total quantity) and *the predicted clearing price* (classified in one of 24 states where the best (lowest) prices are in state 1 and the highest price in state 24).
- ***Actions:*** Remaining energy ratio (ratio of remaining energy that needs to be ordered i.e. 10, 20 or 80 %) and bid price (which actually comes from the clearing price predictor).

- **Transition:** Probability ($P(s'|s, a)$) that the AstonTAC MDP transitions from one state $s \in S$ to another $s' \in S$ after taking the action $a \in A$.
- **Reward:** There are two types of rewards:
 - an immediate reward for buying at a low price, e.g. if the predicted clearing price is high, the agent will receive a positive reward for buying a low energy volume and a negative reward for buying a high energy volume.
 - a delayed reward for the energy balancing after 24 decision steps, the highest positive reward is received when the imbalance is the lowest.

To solve this MDP and find the optimal value function and optimal policy, AstonTAC requires the model parameters (reward function, expected payoffs, and transition probabilities). It learns this parameter online by using a technique inspired by Monte Carlo methods. During the simulation, AstonTAC uses the pay-off average to decide the optimal action to take instead of computing the transition probabilities and the expected reward values. The learning takes place once the information on the energy imbalance is available at the end of each episode (24 timeslots).

Bidding in Periodic Double Auctions Using Heuristics and Dynamic Monte Carlo Tree Search [80, 81]

In this paper, the authors present the details of their broker, SPOT, which primarily consists of a wholesale module for the PowerTAC wholesale PDA. The bidding strategy for PDAs is based on predicting auction clearing prices using **Monte Carlo Tree Search (MCTS)** and using it to plan for current as well as future timeslots. They also showcase the way to search the continuous space of clearing prices in the PowerTAC environment using the systematic search approach.

The MCTS strategy needs an efficient market-clearing price forecaster for current as well as future timeslots. The authors propose two different methods for the task, (i) MDP Price Predictor and (ii) RepTree Price Predictor. The MDP price predictor of SPOT is

implemented by using the wholesale module of TacTex'13, which is explained above. The RepTree price predictor uses ***Reduced Error Pruning (REP)*** to build a supervised decision tree [82]. The RepTree price predictor is finalized as it proved to be more efficient than the MDP price predictor. Note that, in the MCTS method, two important steps are simulation and rollout. SPOT mimics the PowerTAC market clearing mechanism to do the MCTS simulation step. For the rollout step, when the broker reaches to a state where there are no children, it randomly selects actions and appends a state to the tree. The main components of the MCTS strategies are presented below.

- ***States:*** Each node of the MCTS tree describes the state. The state contains *action-id* that led to the current state, *visit-count*, and *avgUnitCost* incurred by the broker in this auction and all future auctions.
- ***Actions:*** The vertices of the MCTS tree denote the actions. Each action is represented by $\{\mu, \sigma, \{\Delta_{min}, \Delta_{max}\}, \gamma\}$, where μ is the predicted limit price, σ is observed standard deviation in clearing prices, $\{\Delta_{min}, \Delta_{max}\}$ are the minimum and maximum price multipliers, γ is volume multiplier. γ decides what percentage of volume to bid out of the remaining demand. Δ_{min} and Δ_{max} get multiplied with σ and then added to μ to create bid range μ_{min} to μ_{max} . SPOT places multiple equidistant bids between bid range μ_{min} to μ_{max} .
- ***Transition:*** A state S_m^n transitions to one of the states in the next time period, $S_0^n - 1, \dots, S_m^n - 1$. Here, n denotes the hour-ahead, and m denotes the action.
- ***Terminal States:*** The states with $n = 0$ (zero hour-ahead states) are the terminal states, S_0^0, \dots, S_m^0
- ***Reward:*** There are two types of rewards:
 - If, during rollout or simulation, the broker reaches a terminal state, then the reward is the balancing cost.

- Otherwise, the reward is the simulation cost, that is, the cost paid for electricity in all the auctions during rollout or simulation.

Basically, SPOT employs an MCTS-based bidding strategy along with several heuristic techniques to optimize bid prices and strategically place multiple bids in auctions. Specifically, it calculates the standard deviation of clearing prices σ offline and incorporates an external limit price predictor that provides limit-price μ . Utilizing a discrete action space, each action includes two multipliers for the limit-price (Δ_{min} and Δ_{max}) and a fraction γ for bid quantity. MCTS selects an appropriate action, and multiple bids are strategically placed within the price range of $\mu + \Delta_{min} * \sigma$ and $\mu + \Delta_{max} * \sigma$ to procure a total fraction of γ of the remaining quantity.

The MCTS-based bidding strategy proposed by the SPOT broker is one of the first bidding strategies to combine the precision of tree search and the generality of random sampling. It uses an MCTS-based strategy coupled with heuristics on top of the limit price derived from a REPTree-based limit price predictor to determine the optimal bid price while placing multiple bids in each auction instance by equally distributing the remaining required quantity among all the bids.

Bidding in Smart Grid PDAs: Theory, Analysis and Strategy [26, 28]

In this paper, the authors present the architecture of the broker VidyutVanika18. Here, we talk about VidyutVanika18's wholesale market module. It uses an MDP-based bidding strategy, where the MDP is solved with the help of dynamic programming and limit-price predictor. The MDP suggests a suitable limit price for each of the 24 auction instances. Additionally, it uses a heuristic strategy to spread the bid quantity across 24 instances based on the predicted limit prices.

VidyutVanika18's MDP formulation is largely similar to TacTex13's MDP described above. The main difference is in the calculation of clearing probability $p_{cleared}$ as shown below.

$$p_{cleared}(s, lp) = \frac{\sum_{tr \in trades[s], tr.lcp < lp} tr.cleared_electricity}{\sum_{tr \in trades[s]} tr.cleared_electricity} \quad (4.2)$$

Unlike TacTex13, VidyutVanika18 uses the *last-clearing-price* (*lcp*) in the numerator to determine the *p_{cleared}*. In each cleared auction, the *last-clearing-price* (*lcp*) for bids (asks) is higher (lower) than or equal to the clearing price of the auction; thus, using *lcp* instead of *cp* would provide a better *p_{cleared}* estimation. As *lcp* is unknown in the market, VidyutVanika18 uses dummy bids/asks to determine the *lcp* in the auctions, where it places multiple dummy bids/asks (where quantity is negligible) with bid prices varying in the range of the original bid's limit-price and balancing-price.

Other than that, unlike most other brokers, VidyutVanika18 does not place all the quantity in a single bid; instead, it spreads the bid quantity across all auction instances. It aims to buy higher quantities when the prices are expected to be low and vice-versa.

Monte Carlo Tree Search in Continuous Action Spaces with Execution Uncertainty [83]

In this paper, the authors introduce a novel MCTS algorithm tailored for continuous action spaces with inherent execution uncertainty. The algorithm addresses scenarios where action execution is unpredictable due to limited human or robotic skills and the stochastic nature of real-world environments. It builds on the concept of SPW (Section 3.5.2), where the action space begins small and is expanded iteratively to include high-quality actions. The key innovation proposed in this work is the idea that samples of execution uncertainty from a particular action provide information about any action that could have generated that execution outcome. This allows for effective generalization across the action space, overcoming a limitation of standard MCTS, which only learns about the specific actions chosen. This generalization is crucial for continuous action spaces, where evaluating every possible action is impractical. To achieve this, the authors propose a new variant called **KR-UCT (Kernel Regression UCT)**.

In KR-UCT, the algorithm moves beyond just evaluating a discrete set of actions by considering the entire continuous action space. By leveraging kernel regression, it generalizes action value estimates across the entire parameter space, facilitating information sharing among all potential actions. This approach enhances the learning process by allowing value information obtained from one action to inform the estimates of others, thus efficiently navigating the continuous space. The details of the KR-UCT algorithm are outlined in Algorithm 10.

In the algorithm, $\mathbb{E}(v|a)$ is the kernel regression function and $W(a)$ is the kernel density function, given by the following equations,

$$\mathbb{E}(v|a) = \frac{\sum_{b \in A} K(a, b) \bar{v}_b n_b}{\sum_{b \in A} K(a, b) n_b}$$

$$W(a) = \sum_{b \in A} K(a, b) n_b$$

The KR-UCT algorithm adapts the standard UCT equations by substituting the traditional exploitation and exploration terms with kernel regression and kernel density terms. The kernel regression function, denoted as $\mathbb{E}(v|a)$, estimates the expected value at a given point by averaging the values of all points in the dataset, weighted according to the kernel function $K(a, b)$. This function determines the influence between pairs of points, reflecting the probability of executing action b when action a is intended, thereby capturing execution uncertainty. The kernel density function $W(a)$ assesses the volume of relevant data available at a particular point, indicating how well-explored that area is. As depicted in the algorithm, KR-UCT leverages kernel regression to facilitate information sharing among actions with similar outcome distributions. This mechanism not only enriches value estimation across the continuous space but also strategically directs the exploration towards actions beyond the initial set of candidates.

KR-UCT

Algorithm 10: KR-UCT($state$)

Input: $state$ node

Output: Rollout value rv , boolean $isTerminal$

```

1: if  $state$  is terminal then
2:   return utility( $state$ ), false
3: end if
4:  $expanded \leftarrow$  false
5:  $A \leftarrow$  actions considered in  $state$ 
6:  $action \leftarrow argmax_{a \in A} \mathbb{E}(v|a) + C \sqrt{\frac{\log \sum_{b \in A} W(b)}{W(a)}}$ 
7: if  $\sqrt{\sum_{a \in A} n_a} < |A|$  then
8:    $newState \leftarrow$  child of  $state$  by taking  $action$ 
9:    $rv, expanded \leftarrow$  KR-UCT( $newState$ )
10: end if
11: if  $!expanded$  then
12:    $newAction \sim argmin_{K(action,a) > \tau} W(a)$ 
13:   add  $newAction$  to  $state$ 
14:    $newState \leftarrow$  child of  $state$  by taking  $newAction$ 
15:   add initial action to  $newState$ 
16:    $rv \leftarrow$  ROLLOUT( $newState$ )
17: end if
18: Update  $\bar{v}_{action}$ ,  $n_{action}$ , and KR with  $rv$ 
19: return  $rv$ , true

```

Other Related Work for Monte Carlo Tree Search in Continuous Action Spaces

Some attempts have been made previously to extend the vanilla MCTS to continuous action spaces. One of the early works is done in HOO [84], which deals with continuous arms by utilizing a tree of coverings of the action space and is inspired by the bandit framework. HOOT [85] improvises on vanilla MCTS by replacing the UCB1 action selection rule with HOO. Alternatively, HOLOP [86] embraces a different strategy by modelling this as a continuous bandit problem, where actions correspond to plans, leveraging HOO. Alternative strategies, such as the progressive widening method, have also been expanded to address stochastic continuous state and action planning predicaments by employing double progressive widening [87], which applies progressive widening to both states and actions. To amplify the efficacy of MCTS integrated with progressive widening, cRAVE [88] integrates the RAVE heuristic [89] using Gaussian convolution, thereby promoting information exchange among actions throughout the entire subtree. Additionally, KR-UCT [90] is another notable approach fostering information sharing among actions within the same node through kernel regression, and it generates new actions guided by kernel density estimation. KR-UCT has showcased superior performance compared to cRAVE in simulated curling environments, earning its acknowledgement as state-of-the-art. The work by Couetoux[91] provides an excellent overview of the MCTS literature.

MCTS-based strategies have also found application in the domain of simultaneous ascending auctions [92, 93], negotiations [94] and sponsored search auctions [95]. Apart from these, MCTS-based strategies have contributed to the domain of PDAs as well, and many interesting MCTS strategies are proposed in the framework of the Power Trading Agent Competition (PowerTAC) [62]. This competition, which closely simulates real-world smart grid scenarios, hosts an annual tournament attracting various teams who deploy broker agents equipped with bidding strategies for energy procurement from the wholesale market PDAs. Notably, SPOT [81], a prominent broker agent in the PowerTAC literature, introduced an MCTS-based bidding strategy augmented with heuristic techniques. In-

spired by the success of SPOT’s MCTS strategy, Tuc-Tac [96] developed its own MCTS bidding strategy, which bears significant resemblance to SPOT’s approach. However, both strategies operate within discretized action spaces, crucial for determining bid prices and quantities in auctions. Specifically, they employ bid price predictors to anticipate suitable bid prices for all auctions. These predicted bid prices serve as input for MCTS, which subsequently determines multipliers and quantity fractions from a discrete set of actions. These multipliers are then applied to the predicted bid price to establish lower and upper bounds for bid prices, within which they place multiple bids for the selected bid quantity. However, all these methods are either limited by the discrete action space or curated for specific problem settings and may not be directly extendable to other problem settings.

Baseline Strategies

- ***Zero Intelligence (ZI):*** The ZI strategy follows a randomized approach to bid in a PDA by ignoring the state of the market; it samples a price from a uniform distribution between the minimum bid price and maximum bid price. Following the ZI strategy, the broker places one bid per auction and the remaining quantity as bid quantity for all 24 auction instances.
- ***Zero Intelligence Plus (ZIP):*** The ZIP agent [97] maintains a scalar variable m denoting the profit it aims to achieve, which gets combined with a unit limit price to compute a bid price p . Small increments adjust the price for each trade with the help of a δ by comparing the submitted bid price and the clearing price. We decide the bid price μ randomly at the start of the game. The profit margin m is set to -1% of μ , resulting in the initial bid price being $p = \mu * 0.99$.

Other than the above-mentioned work, many other strategies have been proposed for bidding in the PowerTAC wholesale market. The CrocodileAgent [98, 99, 100] predicts the demand of the customers for the next 24 timeslots and estimates the wholesale market clearing prices by using a triple exponential smoothing technique, also known as the

Holt-Winters Method. The Mertacor [101, 102] broker uses an adaptive price formation policy and a time series-based policy to forecast customer demand. The bidding strategy of Mertacor is rule-based. Ozdemir and Unland [103] use an adaptive Q-learning-based strategy in their broker AgentUDE. The Maxon [104] broker uses a linear regression model to predict customer demand and split the order into multiple bids at different limit prices. The limit prices are decided based on the last clearing prices. The CWIBroker [105, 106] aims to procure all the required demand in the first auction round, and then it bids for uncleared demand in the remaining rounds. It uses a heuristic to compute its limit price, the limit price for the next timeslot is computed from the worst-case limit price in the most recent timeslot. TUC_TAC [96] too uses an MCTS-based bidding strategy inspired by SPOT [81] in the wholesale market.

4.4.3 In the PowerTAC Tariff Market

Similar to the PowerTAC wholesale market, the tariff market presents an intriguing challenge to designing tariff contract generation strategies. It is a challenge because, unlike wholesale bids, the tariff information is common knowledge in the market, and thus, each broker knows about its opponents' tariffs. Thus, each broker needs to constantly update its tariff to remain competitive in the market. Below, we present some of the widely accepted tariff-generation strategies in PowerTAC, followed by a brief survey of other related tariff-generation strategies.

TacTex'13: A Champion Adaptive Power Trading Agent [77]

As discussed in Section 4.4.2, TacTex aims to design the solution by dividing the problem into two optimization problems, which are being approximated by TacTex's Wholesale Market Module (approximates problem (i)) and Tariff Market Module (approximates problem (ii)). TacTex approximates the above two optimization problems individually. The wholesale market module of TacTex is already presented above. Here, we present its tariff market module.

In the tariff market module, TacTex employs a smart heuristic strategy to select fixed-price tariffs from a set of tariffs. It estimates the utility of each of the tariffs using *EstimateUtility()* function and selects the one having the highest estimated utility. This *EstimateUtility()* function takes into account the subscription change based on a given tariff (using Locally Weighted Linear Regression) and calculates the net utility (*totalIncome-totalCost*) by taking into account income and cost of all the customers.

In the tariff market, TacTex optimizes future demands and selling prices given the predicted energy costs for any given timeslot. The following steps are repeated for every tariff publication timeslot. It starts by generating a set of 100 fixed-rate candidate tariffs and then follows the algorithm. Then the algorithm calls *EstimateUtility()* function for each of these candidate tariffs. Here, the *EstimateUtility()* works by estimating long-term income and costs after publishing a candidate tariff. To calculate this, it first estimates the subscription of a tariff with the use of the Locally Weighted Linear Regression method, which predicts the subscribed population size for a candidate tariff based on its expected rate value. After calculating the subscription, the total income and total cost of the tariff are determined using demand and cost predictor methods. TacTex considers the look-ahead of 1 week ($24*7=168$ timeslots), chosen as a trade-off between shorter horizons, which might not capture weekly consumption/production patterns, and longer horizons, which present higher uncertainty and require more computation. Based on the calculated total income and total cost of a tariff, the algorithm finally compares the utility values of all 100 candidate tariffs and publishes the tariff having the highest utility.

The novelty of this work is that while calculating the utility of any tariff using *EstimateUtility()* function in the retail market, TacTex also considers the effect of subscription change when that tariff is published in the market, which gives a more realistic estimation of the tariff utility.

VidyutVanika: A Reinforcement Learning Based Broker Agent for a Power Trading Competition [28]

Past brokers like COLD Energy and VidyutVanika18, as well as Reddy & Veloso, modelled the decision process in the retail market as an MDP to generate tariff contracts [107, 108, 28]. In fact, both COLD Energy and VidyutVanika18's tariff strategies were motivated by Reddy & Veloso. In these approaches, the state space is constituted by market parameters such as market rationality, agent's portfolio status, etc., and action space was designed with actions to increase or decrease the rate value of tariffs by a certain amount. The reward function was profit in the market. More details about the MDP are provided below,

- **States:** The continuous state of the environment is discretized into the below tuple to represent the state space of the MDP, $\{PRS_t, PS_t, CPS_t, PPS_t\}$, where,

$PRS_t = \{rational, inverted\}$ determines the price range status of the broker. The broker is *rational* if its minimum consumption tariff rate is higher than the maximum production tariff rate; otherwise, it is *inverted*.

$PS_t = \{shortsupply, balanced, oversupply\}$ captures the portfolio status of the broker. If the supply exactly matches the demand, then the status is *balanced*. Otherwise, if the supply is higher than the demand, then the status is *oversupply*; else, the status is *shortsupply*.

$CPS_t = \{out, near, far, veryfar\}$ encodes how far the broker's consumption tariff price is compared to the minimum and maximum consumption prices in the market.

$PPS_t = \{out, near, far, veryfar\}$ encodes how far the broker's production tariff price is compared to the minimum and maximum production prices in the market.

- **Actions:** The broker has 8 actions to increase or decrease the current tariff prices by different step values, as shown below,

$$\{maintain, lower, raise, inline, revert, minmax, wide, bottom\}$$

- **Transitions:** The broker transitions to the next state depending on the market parameters.
- **Terminal State:** This is infinite horizon MDP; thus, there is no terminal state.
- **Reward:** The reward of VidyutVanika18 is different than that of COLD Energy and is given below,

$$r_t = \theta_{t,C} P_{t,C} - \theta_{t,P} P_{t,P} - \theta_{t,W} P_{t,W}$$

Where $\theta_{t,C}$ and $\theta_{t,P}$, are the consumed electricity amount and produced electricity amount in the tariff market, whereas $\theta_{t,W} = \theta_{t,C} - \theta_{t,P}$ is electricity amount procured from the wholesale market. Similarly, $P_{t,i}$ s denote the respective prices to buy/sell electricity. Thus, the reward function calculates the net profit of the broker in the market by considering revenue in the tariff market and cost in the wholesale market.

The MDP is solved using the Q-learning approach, which outputs an FPT. This FPT is then converted to a ToU tariff using the demand predictor module that increases the rate values when demand is expected to be high and vice-versa. Thus, the generated ToU tariff helps mitigate the peak demand penalties by incentivizing customers to shift their usages away from peak timeslots.

Aiming for Half Gets You to the Top: Winning PowerTAC 2020 [96]

The experimental evidence suggests that the seemingly optimal class of strategies of capturing all the market share may suffer from high peak demand penalties as all the customers are subscribed to one agent, and that agent alone has to bear the total penalty for the grid imbalance. To remedy this, agent TUC-TAC proposed a strategy aimed at acquiring only half the retail market share. TUC-TAC takes inspiration from the Lemonade Stand Game (LSG), where N lemonade vendors have to choose a location to set up their counter on the perimeter of a circular island. The utility of each vendor is determined by the distance between his, the neighbour vendors, and the defined space boundaries. The optimal strategy

of this game is that one should always sit opposite to some opponent [109]. To translate this in the PowerTAC setting, one should aim to claim a large enough market share in the tariff market without being too greedy. Having half of the market share would ensure that the broker would get half of the total profits, but only half of the peak demand penalties would be charged to the broker.

The main tariff strategy of TUC-TAC is heuristics-based, where it aims to maintain half of the market share during the games. More specifically, it keeps two hyperparameters, Middle-bound and Lower-bound. These hyperparameters help it to maintain the market share close to 50%. If, at any time, the market share is higher than Middle-bound, then it instantly revokes its cheapest tariff and generates a new tariff in order to bring the market share down to Middle-bound. Similarly, if the market share is lower than Lower-bound, then it attempts to generate a tariff that is more attractive for the customers to increase the market share.

Similar to PowerTAC wholesale strategies, a number of strategies for the PowerTAC tariff market have been developed by past participants. CrocodileAgent [98, 99, 100] keeps only profitable and higher utility tariffs; unprofitable tariffs and tariffs that show insufficient utility over time are revoked periodically. It offers ToU tariffs generated based on pre-determined rate values. The Mertacor [101, 102] solves optimization problems in the tariff market with the aim of maximizing the broker's profit and retaining an adequate customer market share. It uses Particle Swarm Optimization (PSO) techniques to solve these optimization problems. AgentUDE [110] utilizes online genetic algorithm-based optimization techniques and aggressive pricing to design ToU tariffs that also contribute to reducing peak demand charges. Maxon [104] uses different kinds of tariffs for different types of customers in the tariff market. Tiered and TOU tariffs enable the flattening of peak demands by rewarding customers for altering their usage behaviour. Maxon uses a technique to scale the prices of the maximum and minimum demands using the baseline price. The scaling factors are determined by an inner Hill Climbing Algorithm. CWIBroker [105, 106]

broker uses a heuristic inspired by the Tit-For-Tat strategy in Iterated Prisoner's Dilemma to determine tariff rates based on competing tariffs.

Many other approaches in the literature have been suggested to tackle the tariff generation problem, and a few have been implemented in PowerTAC as well. In the retail market of smart grids, techniques such as demand response, peak demand pricing, and learning-based approaches have been proposed to design competitive tariffs. Many multi-armed bandit-based strategies have been proposed to publish tariffs in the smart grid domain. Most of this work focuses on demand response in a smart grid where customers are incentivized via tariffs to curtail their usages in response to electricity supplier's signals [111, 112, 113, 114, 115, 116].

4.4.4 Demand Response for Peak Reduction

In this section, we present peak detection and reduction strategies, specifically focusing on demand response-based strategies. In the PowerTAC, the peak reduction strategies are mainly implemented via offering ToU tariffs with incentives to shift usage away from peak timeslots, which is also a form of demand response. However, there is a rich literature on peak reduction techniques using demand response that aims to find suitable customers for the task and design mechanisms to offer them appropriate incentives. Below, we present some of the useful techniques.

Robust Peak Detection Algorithm Using Z-Scores [117]

Here the problem of detecting the peaks in the real-time time-series data is addressed. The proposed algorithm detects the peaks and troughs (negative peaks) in real-time data by taking into account the knowledge of newer data points as well as the history. The proposed algorithm uses a z-score to detect the peaks. The z-score is the signed fractional number of standard deviations by which the value of a data point is above the mean value of what is being observed or measured, $z = \frac{x-\mu}{\sigma}$; here, μ is the running mean and σ is the running standard deviation (sd). z indicates by how many standard deviations the data point x is

away from the mean. This running mean and running SD are calculated over a specific lag of filtered data points. The lag parameter determines how much data will be smoothed and how adaptive the algorithm is to changes in the long-term average of the data. The author maintains a list of filtered data points where instead of directly including the peak data point in the list, he lets the user decide the influence of this peak data point on the running mean and sd. The influence (ranges between 0 to 1) controls the robustness of the algorithm; a peak data point with influence 0 does not influence the running mean and running sd at all, and the future z-scores are calculated as if this peak had not happened. The influence value of more than 0 affects the running mean and running sd, and thus future z-scores. The peak is detected when the absolute value of the z-score is higher than a specified threshold. This threshold is a pre-defined value that decides the tolerance level.

DR involves brokers offering customers monetary incentives to optimize their electricity load voluntarily. There are many approaches, such as auction-based mechanisms [118, 119] and dynamic pricing [120] to achieve DR. The major challenge with these approaches is that different customers may respond differently to the given incentives. Thus, to increase customer participation, it becomes crucial to learn their reaction toward these incentives. Learning customers' behaviour is challenging due to the uncertainty and randomness that creep in due to exogenous factors like weather [112, 116].

Works like [112, 116] consider a very simplistic model - when a broker offers a customer incentive more than what it values, the customer reduces every unit of electricity it consumes with a certain probability independent of the incentive. This probability is termed as RP [111, 112]. RPs are learned using MAB solutions. There are three primary issues with these approaches. In these approaches, customers' valuations need to be elicited [111, 112], which adds additional communication complexity. Some of the other popular ones include ToU tariff [121, 122], direct load control [123], the price elasticity of demand approach (dynamic pricing) [124] approaches. These approaches are quite complex for the customers as the price keeps changing. It can lead to customer confusion due to uncertain supply,

volatile prices, and lack of information. Due to the complexity involved in these methods, many recent works have focused on providing incentives to the customers, which make them shift their load from peak hours to non-peak hours [125, 111].

In the literature, many techniques for providing incentives primarily focus on the setting where when given an offer (incentive), the consumer can either reduce or choose not to reduce the consumption. For example, the DR mechanism in [111, 112] considered a setting where each consumer was associated with two quantities: (i) valuation per unit of electricity, which represents how much a consumer values the unit of electricity, and (ii) acceptance rate, which denotes the probability of accepting the offer if a consumer is given the incentive more than his/her valuation. The authors then proposed a Multi-Armed Bandit mechanism that elicits the valuation of each consumer and learns the acceptance rate over a period of time. Similar approaches were also considered in [113, 114, 115, 116]. Two sets of works aim to maximize the peak reduction under a budget constraint. (i) With a mixed integer linear programming (MILP) approach [126], and (ii) with an efficient algorithm by drawing similarities from the min-knapsack problem [127].

4.5 Summary

In this chapter, we presented the details of the problem domain, the smart grids. To validate any novel smart grid strategy, one may need an efficient simulator, and PowerTAC provides one such simulator. We described the PowerTAC simulator comprehensively, along with details of all three PowerTAC markets. Then, we highlighted the broker's activities in the simulator in all three markets. In the end, we presented a thorough review of past strategies proposed in the literature. Having established a solid foundation, we now shift our focus to presenting our key contributions in the subsequent chapters.

Part A: The Wholesale Market

Designing Bidding Strategies in Wholesale

Market

Chapter 5

Designing Bidding Strategies via Equilibrium Theory

Periodic double auctions (PDA) have applications in many areas, such as in e-commerce, intra-day equity markets, and day-ahead energy markets in **smart-grids**. While the trades accomplished using PDAs are worth trillions of dollars, finding a reliable bidding strategy in such auctions is still a challenge as it requires consideration for future auctions. A participating buyer in a PDA has to design its bidding strategy by planning for current and future auctions. Many **equilibrium-based bidding strategies** proposed are complex to use in real-time. In this chapter, we propose a scale-based bidding strategy for buyers participating in PDA. We first present an equilibrium analysis for **single-buyer single-seller multi-unit single-shot k -Double auctions**. Specifically, we analyze the situation when a seller and a buyer trade n identical units of quantity in a double auction where both the buyer and the seller deploy a simple, scale-based bidding strategy.

The equilibrium analysis becomes intractable as the number of participants increases. To be useful in more complex settings, such as wholesale markets

in smart grids, we model equilibrium bidding strategy as a learning problem.

We develop a **Deep Deterministic Policy Gradient (DDPG)**-Based Bidding Strategy, **DDPGBBS**, for a participating agent in PDAs to suggest an action at any auction instance. *DDPGBBS*, which empirically follows the obtained theoretical equilibrium, is easily extendable when the number of buyers/sellers increases. We take PowerTAC’s wholesale market PDA as a testbed to evaluate our novel bidding strategy. We benchmark our DDPG-based strategy against several baselines and state-of-the-art bidding strategies of the PowerTAC wholesale market PDA and demonstrate the efficacy of *DDPGBBS* against several benchmarked strategies¹.

5.1 Introduction

An auction is a process of buying or selling goods or items [128], and it could be of varied types [16]. For example, a **forward auction** consists of one seller and multiple buyers who submit their **bids**; a **reverse auction** has one buyer and multiple sellers submitting their **asks**; a **double auction** comprising of multiple buyers and sellers placing their bids and asks. Double auctions are extensively used in the real world to trade stocks, energy, spectrum, and many other goods and services. Stock Exchanges are one example where a double auction generally takes place and involves trillions of dollars in daily trades. Among all the domains, **smart-grids** are the prominent domain where the double auction plays a significant role [11].

As discussed in Chapter 4, GenCos and energy brokers trade energy in the wholesale market through double auctions. Nord Pool, which runs the leading power market in Europe, showed trades of 995 *TWh* of volume, with close to 60% of the volume traded using APIs [10]. Clearly, a bidding strategy that can optimize the cost of energy brokers even by a small amount would significantly improve their profits. As discussed, the setup of periodic

¹The preliminary work has been published at AAMAS 2022, and the extended work is under review at AIJ.

double auctions (PDAs) facilitates the periodic trading of commodities before the actual delivery slot. While such PDAs are common in real-world markets, devising optimal real-time bidding strategies for brokers remains challenging due to the sequential nature of these auctions. Players must strategically plan their bids and asks across multiple auction rounds, considering both current market conditions and future uncertainties, e.g., in day-ahead auctions—a prevalent form of PDAs—a buyer can secure commodities for a specific delivery hour within the preceding 24 hours. Let us take an example of PowerTAC wholesale market auctions; an energy broker participating in day-ahead markets can procure energy for a delivery timeslot at any prior time within 24 hours; this scenario is common for real-world smart-grid auctions as well.

Motivated by the literature where an equilibrium strategy often takes the form of a scalar multiplication function of the valuation ($f(v) = k \cdot v$) [14], we explore ***scaling-based equilibrium bidding strategies*** for the following reasons: - (i) the simplicity such strategies offer, (ii) they are the natural when the brokers need to submit bids in real-time, and (iii) as compared to solving differential equations [12] or fictitious play formulations [13] to devise strategies (which are complex), scale-based strategies are straightforward to implement and befit for real-time bidding in the settings such as smart grids. Our analysis starts with a study of a multi-unit, SSDA involving a single seller and a single buyer, strategically employing scaling-based strategies. In this setup, the buyer and seller operate under imperfect information, where they do not have visibility into each other's bids. This scenario is modelled as a two-player general sum game, where we characterize Bayesian Nash equilibria (BNE) for both players trading multiple units of identical, indivisible items. However, characterizing equilibrium solutions becomes increasingly complex as the number of players grows or when transitioning to PDAs. To address this, we explore whether an intelligent agent (buyer) can ***empirically*** learn the equilibrium bidding strategy for PDAs.

We use techniques from ***RL*** and, more specifically, we develop a DDPG-based bidding strategy to address the issue of learning equilibrium strategy. The strategy thus developed

is then tested on the PowerTAC wholesale market. We chose DDPG, as it has proved to be very successful in a variety of *planning* tasks [55]. *DDPGBBS* learns to bid optimally with the help of feedback from the auction environment. To the best of our knowledge, we are the first to utilize the policy gradient-based RL algorithm, enabling us to work with PowerTAC’s continuous state and action space more effectively.

To test the efficacy of *DDPGBBS* to *learn equilibrium strategies*, first, we conduct validation experiments to confirm if it learns the known equilibria. We observe that it can approximate the NE under the controlled experimental setting for all the known cases. We designed a simulator comprising one buyer, one seller, and a two-unit double auction to perform validation experiments. We model the seller to follow its NE strategy and train the agent (buyer) against it.

Post verifying the ability to learn equilibrium bidding strategy with *DDPGBBS*, we extend it to model general PDA where there is no restriction on the number of participants or the number of units traded in the auction. As there are multiple opportunities for the buyer to buy energy for a particular delivery timeslot, we introduce *proximity* to the delivery timeslot into our state-space formulation. For training and validating *DDPGBBS* for such real-world scenarios, we leverage the PowerTAC PDA simulation environment to conduct all our experiments.

5.2 Preliminaries

In this section, we formally define the auction setting under analysis and introduce the notations used in this chapter as well as in the later chapters. We consider a market with B buyers competing to procure multiple units of a commodity from S sellers by participating in a PDA with H rounds. In any round $h \in [H]^2$ of the PDA, a buyer $b \in [B]$ has $Q^{b,h}$ units of a commodity to purchase, with each $Q^{b,h}$ being unique for every buyer $b \in [B]$. The set of bids for buyer b in round h is denoted as $\mathcal{B}^{b,h}$, containing $|\mathcal{B}^{b,h}|$ elements.

²We denote $[K]$ as the set $1, \dots, K$ for any integer K .

Each bid is represented as a pair of price and quantity $(p_i^{b,h}, q_i^{b,h})$, where $i \in [|\mathcal{B}^{b,h}|]$, with the price and quantity bounded by p_{\max} and $Q^{b,h}$, respectively. Consequently, the total outstanding demand in round h from all B buyers is given by $Q^{\mathcal{D},h} = \sum_{b \in [B]} Q^{b,h}$. Furthermore, we assume that the buyers cannot purchase more units of a commodity than their requirement.

Conversely, the sellers are represented by a consolidated supply curve with $|\mathcal{L}^h|$ asks, expressed as $\mathcal{L}^h = \{(p_i^h, q_i^h); i \in [|\mathcal{L}^h|]\}$, where $p_i^h \in [0, p_{\max}]$ and $q_i^h \in [0, q_{\max}]$ are the price and quantity components of the i^{th} ask (p_i^h, q_i^h) , with p_{\max} and q_{\max} serving as suitable upper bounds. The total supply provided by wholesale sellers in round h is denoted as $Q^{\mathcal{S},h} = \sum_{i \in [|\mathcal{L}^h|]} q_i^h$.

At each round h , the market regulator collects the elements of \mathcal{L}^h and \mathcal{B}^h and applies a *clearing rule* to determine the clearing price(s) and quantities. This work focuses on *uniform pricing* as the payment rule, where all cleared bids and asks have the same clearing price. The cleared price for round h is referred to as the market clearing price, denoted by λ^h , and the total quantity cleared for all the $|\mathcal{B}^h|$ bids in round h is denoted as Q^h . To elucidate the clearing process, we assume, without loss of generality, that the elements of the set \mathcal{L}^h (\mathcal{B}^h) are sorted in increasing (decreasing) order of their price components. If the price components are equal, then \mathcal{L}^h and \mathcal{B}^h are sorted in decreasing order of their quantity components. If the price and quantity components are identical, the ordering between them is arbitrary. We adopt uniform pricing as the payment rule, ensuring that all winning buyers and sellers pay or receive the same price per unit of the commodity traded, dependent on the market clearing rule. One of the most prominent clearing rules for k -double auctions is defined in Section 4.2.2.1.

Benchmark Strategies

We use four PDA wholesale bidding strategies to benchmark DDPGBBS performance, namely, ZI, ZIP, VidyutVanika18, and SPOT. The description of all these strategies is available in Section 4.4.2.

5.3 Equilibrium Analysis for Multi-unit Double Auction

We start our analysis with *single-shot multi-unit double auctions*, where commodities are traded in a single auction round. A single-shot auction is a particular case of PDA with $H = 1$. In this simplified scenario, we assume only one buyer and one seller are in the market, each trading for n units of the commodity by placing n bids and asks, respectively (one bid/ask per unit). According to the notations defined earlier, we have $B = 1$, $S = 1$ and $H = 1$.

In this single auction round, the buyer $b \in [B]$ (denoted by b) has $Q^b = n$ units of the commodity to purchase. The buyer places individual bids for each unit, denoted collectively as \mathcal{B}^b with $|\mathcal{B}^b| = n$ elements³. Each bid is represented as a pair of price and quantity, denoted by (p_i^b, q_i^b) , where $i \in \{1, \dots, n\}$. Similarly, the seller (denoted by s) places individual asks for each unit of the commodity, collectively denoted as \mathcal{L}^s with $|\mathcal{L}^s| = n$ elements. Each ask is represented as a pair of price and quantity, denoted by (p_i^s, q_i^s) , where $i \in \{1, \dots, n\}$.

Assume the true types (true valuations) of buyer b and seller s are θ^b and θ^s , respectively. Both players utilize *scale-based bidding strategies* to determine their bids \mathcal{B}^b and asks \mathcal{L}^s . The buyer b employs a scale-based strategy by placing n bids $\mathcal{B}_i^b = (\alpha_i^b \theta^b, 1)$, for $i \in \{1, \dots, n\}$, while the seller s places n asks $\mathcal{L}_i^s = (\alpha_i^s \theta^s, 1)$, for $i \in \{1, \dots, n\}$ by following a similar scale-based strategy. Here, α_i^b and α_i^s are the scale factors by which buyer b and seller s scales its true type, respectively (assuming $\alpha_1^b \geq \alpha_2^b \geq \dots \geq \alpha_n^b$ and $\alpha_1^s \leq \alpha_2^s \leq \dots \leq \alpha_n^s$). We further assume that $\theta^b \sim U[l^b, h^b]$ and $\theta^s \sim U[l^s, h^s]$, which is common knowledge. This implies that the true types of the buyer and seller are sampled from a uniform distribution within the specified intervals. The expected utilities of b and s are denoted by U^b and U^s , respectively.

³As there is only a single buyer and single seller in a single-shot auction, we simplify the notation by omitting the indices of players and auction rounds.

We start with an analysis of the *buyer's expected utility*. There are $n + 1$ possible clearing scenarios. We calculate the utility for each scenario separately and later combine them. u_0^b is the expected utility when no clearing happens, u_i^b is the expected utility when i units are cleared, $i \in \{1, \dots, n\}$. As the market does not clear in the first scenario, $u_0^b = 0$. Figure 5.1 and Table 5.1 illustrate the appropriate conditions and corresponding clearing prices for the remaining scenarios. In Figure 5.1, the blue box in each plot indicates the area where market clearing occurs, which forms the basis for the conditions derived in Table 5.1. The red line represents the possible clearing prices under the k -double auction setting, which is a convex combination of the last cleared bid and ask prices. For the buyer, n units of item gets cleared with a clearing price of $k\alpha_n^b\theta^b + (1 - k)\alpha_n^s\theta^s$ between the interval $l^s \leq \theta^s \leq \frac{\alpha_n^b}{\alpha_n^s}\theta^b$. Similarly, for any other i units of item, where $1 \leq i \leq n - 1$, clearing happens within the interval $\frac{\alpha_{i+1}^b}{\alpha_{i+1}^s}\theta^b \leq \theta^s \leq \frac{\alpha_i^b}{\alpha_i^s}\theta^b$ with a clearing price of $k\alpha_i^b\theta^b + (1 - k)\alpha_i^s\theta^s$. Note that, we are assuming $\alpha_1^s \leq \alpha_2^s \leq \dots \leq \alpha_n^s$. For the utility calculation of the buyer, the valuation is taken as positive as the buyer has positive utility for the commodity, and the payment (which is a clearing price) is negative as the buyer makes the payment. In the following analysis, we present only the final equations to enhance readability.

Table 5.1: Clearing Conditions and Prices for Buyer's Utility

#Units Clearing	Conditions	Clearing Price
1 Unit Clearing	$\alpha_2^b\theta^b \leq \alpha_2^s\theta^s$ and $\alpha_1^s\theta^s \leq \alpha_1^b\theta^b$ $\frac{\alpha_2^b}{\alpha_2^s}\theta^b \leq \theta^s \leq \frac{\alpha_1^b}{\alpha_1^s}\theta^b$	$k\alpha_1^b\theta^b + (1 - k)\alpha_1^s\theta^s$
i Unit Clearing	$\alpha_{i+1}^b\theta^b \leq \alpha_{i+1}^s\theta^s$ and $\alpha_i^s\theta^s \leq \alpha_i^b\theta^b$ $\frac{\alpha_{i+1}^b}{\alpha_{i+1}^s}\theta^b \leq \theta^s \leq \frac{\alpha_i^b}{\alpha_i^s}\theta^b$	$k\alpha_i^b\theta^b + (1 - k)\alpha_i^s\theta^s$
n Unit Clearing	$\alpha_n^s l^s \leq \alpha_n^s\theta^s$ and $\alpha_n^s\theta^s \leq \alpha_n^b\theta^b$ $l^s \leq \theta^s \leq \frac{\alpha_n^b}{\alpha_n^s}\theta^b$	$k\alpha_n^b\theta^b + (1 - k)\alpha_n^s\theta^s$

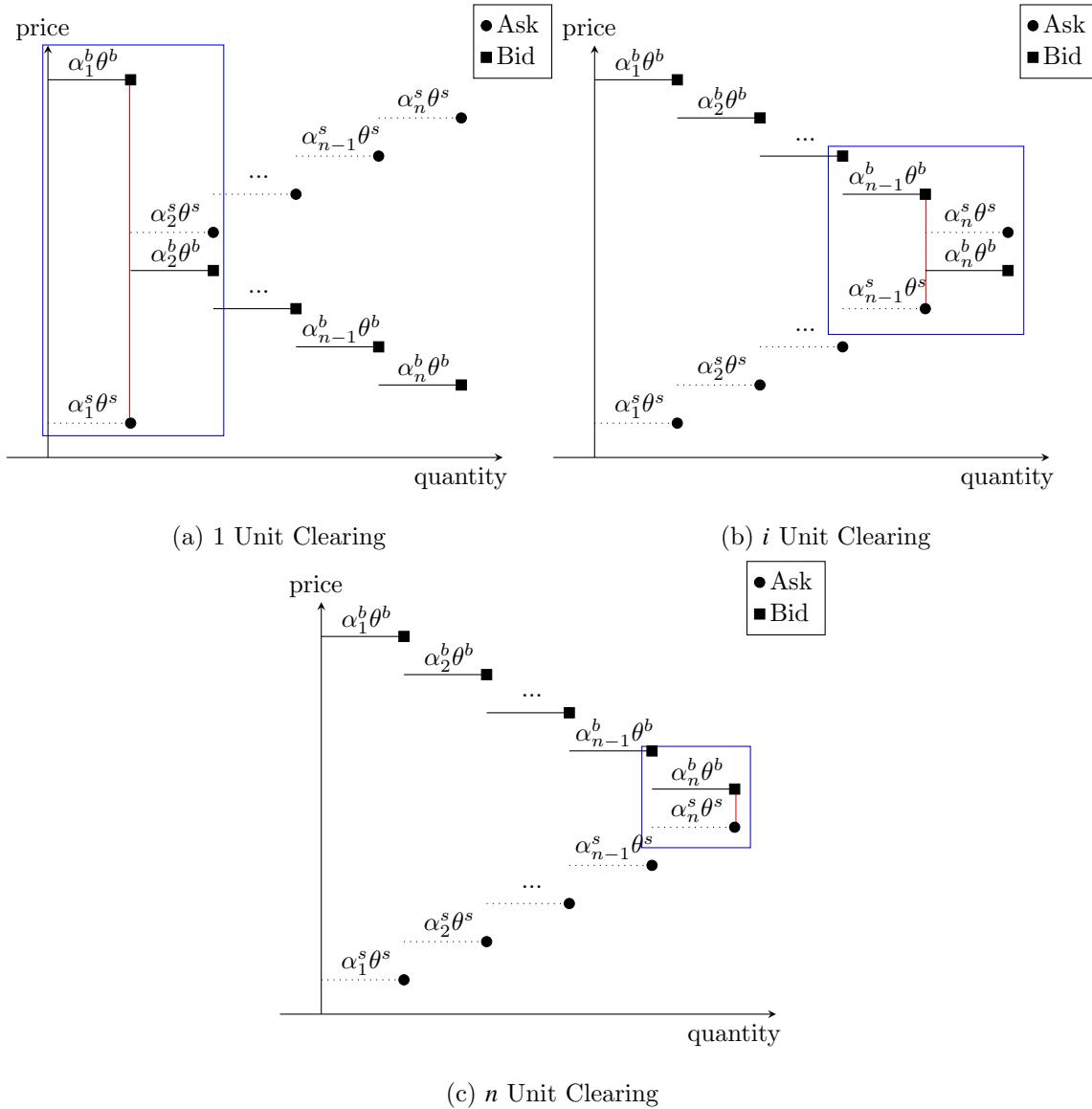


Figure 5.1: Clearing Scenarios for 1 Buyer, 1 Seller and n Units of Items

With the foundational knowledge established, we now proceed to calculate the utilities for each clearing scenario. We begin with the utility for the total clearing case, u_n^b ,

$$u_n^b = n \int_{l^s}^{\frac{\alpha_n^b}{\alpha_n^s} \theta^b} \left(\theta^b - \left(k \alpha_n^b \theta^b + (1-k) \alpha_n^s \theta^s \right) \right) \cdot \left(\frac{1}{h^s - l^s} \right) d\theta^s \quad (5.1)$$

For $i \in \{1, \dots, n-1\}$, the utilities are as following,

$$u_i^b = i \int_{\frac{\alpha_{i+1}^b}{\alpha_{i+1}^s} \theta^b}^{\frac{\alpha_i^b}{\alpha_i^s} \theta^b} \left(\theta^b - \left(k \alpha_i^b \theta^b + (1-k) \alpha_i^s \theta^s \right) \right) \cdot \left(\frac{1}{h^s - l^s} \right) d\theta^s \quad (5.2)$$

Thus, the total utility U^b for the buyer would be,

$$U^b = \int_{l^b}^{h^b} \sum_{i=1}^n u_i^b \cdot \left(\frac{1}{h^b - l^b} \right) d\theta^b \quad (5.3)$$

Equations 5.1 and 5.2 calculate the interim utility of the buyer for the scenarios depending on number of units of clearance. The integral limits in equations 5.1 and 5.2 denote the regions where that many units of clearance happen. As we assume that the true types are sampled from the uniform distribution, the normalizing factors denote the probabilities of sampling from such a distribution. Equation 5.3 calculates the resultant expected utility of the buyer. In order to solve Equation 5.3, we first need to solve equations 5.1 and 5.2. A simple integration and then simplification would lead to the following equations,

$$u_n^b = \left(\frac{n}{h^s - l^s} \right) \left[(1 - k \alpha_n^b) \left(\frac{\alpha_n^b}{\alpha_n^s} \theta^b - l^s \right) \theta^b - \frac{(1-k) \alpha_n^s}{2} \left(\left(\frac{\alpha_n^b}{\alpha_n^s} \theta^b \right)^2 - (l^s)^2 \right) \right] \quad (5.4)$$

For $i \in \{1, \dots, n-1\}$,

$$u_i^b = \left(\frac{i(\theta^b)^2}{h^s - l^s} \right) \left[(1 - k\alpha_i^b) \left(\frac{\alpha_i^b}{\alpha_i^s} - \frac{\alpha_{i+1}^b}{\alpha_{i+1}^s} \right) - \frac{(1 - k)\alpha_i^s}{2} \left(\left(\frac{\alpha_i^b}{\alpha_i^s} \right)^2 - \left(\frac{\alpha_{i+1}^b}{\alpha_{i+1}^s} \right)^2 \right) \right] \quad (5.5)$$

Now, as per Equation 5.3, we need to take a sum over all the individual interim utilities $u_i^b, 1 \leq i \leq n$ using equations 5.4 and 5.5.

$$\begin{aligned} \sum_{j=1}^n u_j^b &= u_1^b + u_2^b + \dots + u_n^b \\ \sum_{j=1}^n u_j^b &= \left(\frac{(\theta^b)^2}{h^s - l^s} \right) \left[(1 - k\alpha_1^b) \left(\frac{\alpha_1^b}{\alpha_1^s} - \frac{\alpha_2^b}{\alpha_2^s} \right) - \frac{(1 - k)\alpha_1^s}{2} \left(\left(\frac{\alpha_1^b}{\alpha_1^s} \right)^2 - \left(\frac{\alpha_2^b}{\alpha_2^s} \right)^2 \right) \right] + \\ &\quad \left(\frac{2(\theta^b)^2}{h^s - l^s} \right) \left[(1 - k\alpha_2^b) \left(\frac{\alpha_2^b}{\alpha_2^s} - \frac{\alpha_3^b}{\alpha_3^s} \right) - \frac{(1 - k)\alpha_2^s}{2} \left(\left(\frac{\alpha_2^b}{\alpha_2^s} \right)^2 - \left(\frac{\alpha_3^b}{\alpha_3^s} \right)^2 \right) \right] + \dots + \\ &\quad \left(\frac{n}{h^s - l^s} \right) \left[(1 - k\alpha_n^b) \left(\frac{\alpha_n^b}{\alpha_n^s} \theta^b - l^s \right) \theta^b - \frac{(1 - k)\alpha_n^s}{2} \left(\left(\frac{\alpha_n^b}{\alpha_n^s} \theta^b \right)^2 - (l^s)^2 \right) \right] \\ \sum_{j=1}^n u_j^b &= \left(\frac{1}{h^s - l^s} \right) \left[(\theta^b)^2 \left(\frac{\alpha_1^b}{\alpha_1^s} - \frac{\alpha_2^b}{\alpha_2^s} \right) + 2(\theta^b)^2 \left(\frac{\alpha_2^b}{\alpha_2^s} - \frac{\alpha_3^b}{\alpha_3^s} \right) + \dots + n\theta^b \left(\frac{\alpha_n^b}{\alpha_n^s} \theta^b - l^s \right) - \right. \\ &\quad k \left(\alpha_1^b (\theta^b)^2 \left(\frac{\alpha_1^b}{\alpha_1^s} - \frac{\alpha_2^b}{\alpha_2^s} \right) + 2\alpha_2^b (\theta^b)^2 \left(\frac{\alpha_2^b}{\alpha_2^s} - \frac{\alpha_3^b}{\alpha_3^s} \right) + \dots + n\alpha_n^b \theta^b \left(\frac{\alpha_n^b}{\alpha_n^s} \theta^b - l^s \right) \right) - \frac{(1 - k)}{2} \\ &\quad \left. \left((\theta^b)^2 \alpha_1^s \left(\left(\frac{\alpha_1^b}{\alpha_1^s} \right)^2 - \left(\frac{\alpha_2^b}{\alpha_2^s} \right)^2 \right) + 2(\theta^b)^2 \alpha_2^s \left(\left(\frac{\alpha_2^b}{\alpha_2^s} \right)^2 - \left(\frac{\alpha_3^b}{\alpha_3^s} \right)^2 \right) + \dots + n\alpha_n^s \left(\left(\frac{\alpha_n^b}{\alpha_n^s} \theta^b \right)^2 - (l^s)^2 \right) \right) \right] \\ \sum_{j=1}^n u_j^b &= \left(\frac{1}{h^s - l^s} \right) \left[(\theta^b)^2 \left(\frac{\alpha_1^b}{\alpha_1^s} \right) + (\theta^b)^2 \left(\frac{\alpha_2^b}{\alpha_2^s} \right) + \dots + (\theta^b)^2 \left(\frac{\alpha_n^b}{\alpha_n^s} \right) - n\theta^b l^s \right. \\ &\quad - \left. \left\{ (\theta^b)^2 \left(\frac{\alpha_1^b}{\alpha_1^s} - \frac{\alpha_2^b}{\alpha_2^s} \right) \left(k\alpha_1^b + \frac{(1 - k)}{2} \alpha_1^s \left(\frac{\alpha_1^b}{\alpha_1^s} + \frac{\alpha_2^b}{\alpha_2^s} \right) \right) + 2(\theta^b)^2 \left(\frac{\alpha_2^b}{\alpha_2^s} - \frac{\alpha_3^b}{\alpha_3^s} \right) \right. \right. \\ &\quad \left. \left. \left(k\alpha_2^b + \frac{(1 - k)}{2} \alpha_2^s \left(\frac{\alpha_2^b}{\alpha_2^s} + \frac{\alpha_3^b}{\alpha_3^s} \right) \right) + \dots + n \left(\frac{\alpha_n^b}{\alpha_n^s} \theta^b - l^s \right) \left(k\theta^b \alpha_n^b + \frac{(1 - k)}{2} \alpha_n^s \left(\frac{\alpha_n^b}{\alpha_n^s} \theta^b + l^s \right) \right) \right\} \right] \end{aligned}$$

$$\sum_{j=1}^n u_j^b = \frac{1}{h^s - l^s} \left[\sum_{j=1}^n \frac{\alpha_j^b}{\alpha_j^s} (\theta^b)^2 - n\theta^b l^s - \sum_{j=1}^{n-1} j(\theta^b)^2 \left(\frac{\alpha_j^b}{\alpha_j^s} - \frac{\alpha_{j+1}^b}{\alpha_{j+1}^s} \right) \left(k\alpha_j^b + \frac{(1-k)\alpha_j^s}{2} \left(\frac{\alpha_j^b}{\alpha_j^s} + \frac{\alpha_{j+1}^b}{\alpha_{j+1}^s} \right) \right) \right.$$

$$- \frac{nk}{\alpha_n^s} (\alpha_n^b)^2 (\theta^b)^2 - \frac{n(1-k)(\alpha_n^b)^2}{2\alpha_n^s} (\theta^b)^2 - \frac{n(1-k)}{2} \alpha_n^b \theta^b l^s + n k \alpha_n^b \theta^b l^s + \frac{n(1-k)}{2} \alpha_n^b \theta^b l^s +$$

$$\left. \frac{n(1-k)}{2} \alpha_n^s (l^s)^2 \right]$$

$$\sum_{j=1}^n u_j^b = \frac{1}{h^s - l^s} \left[\left(\sum_{j=1}^n \frac{\alpha_j^b}{\alpha_j^s} - \sum_{j=1}^{n-1} j \left(\frac{\alpha_j^b}{\alpha_j^s} - \frac{\alpha_{j+1}^b}{\alpha_{j+1}^s} \right) \right) \left(k\alpha_n^b + \frac{(1-k)\alpha_n^s}{2} \left(\frac{\alpha_n^b}{\alpha_n^s} + \frac{\alpha_{n+1}^b}{\alpha_{n+1}^s} \right) \right) \right.$$

$$\left. - \frac{n(1+k)}{2} \frac{(\alpha_n^b)^2}{\alpha_n^s} \right) (\theta^b)^2 + (k\alpha_n^b - 1) n l^s \theta^b + \frac{n(1-k)}{2} \alpha_n^s (l^s)^2 \right]$$

Based on the value of $\sum_{j=1}^n u_j^b$, Solving Equation 5.3 and performing simplification would result in the below final utility of the buyer,

$$U^b = \frac{(h^b)^2 + h^b l^b + (l^b)^2}{3(h^s - l^s)} \left(\sum_{j=1}^n \frac{\alpha_j^b}{\alpha_j^s} - \sum_{j=1}^{n-1} j \left(\frac{\alpha_j^b}{\alpha_j^s} - \frac{\alpha_{j+1}^b}{\alpha_{j+1}^s} \right) \right) \left(k\alpha_n^b + \frac{(1-k)\alpha_n^s}{2} \left(\frac{\alpha_n^b}{\alpha_n^s} + \frac{\alpha_{n+1}^b}{\alpha_{n+1}^s} \right) \right)$$

$$- \frac{n(1+k)}{2} \frac{(\alpha_n^b)^2}{\alpha_n^s} \right) + \frac{(h^b + l^b)}{2(h^s - l^s)} (k\alpha_n^b - 1) n l^s + \frac{n(1-k)}{2(h^s - l^s)} \alpha_n^s (l^s)^2 \quad (5.6)$$

Similarly, we calculate the *seller's expected utility*. Similar to the buyer, there are $n + 1$ possible clearing scenarios. We calculate the utility for each scenario separately and later combine them. u_0^s is the expected utility when no clearing happens, u_i^s is the expected utility when i units are cleared, $i \in \{1, \dots, n\}$. As the market does not clear in the first scenario, $u_0^s = 0$. For the remaining scenarios, Figure 5.1 and Table 5.2 provide the appropriate conditions and corresponding clearing prices. Specifically, n units of item gets cleared with a clearing price of $k\alpha_n^b \theta^b + (1-k)\alpha_n^s \theta^s$ between the interval $\frac{\alpha_n^s}{\alpha_n^b} \theta^s \leq \theta^b \leq h^b$. Similarly, for any other i units of item, where $1 \leq i \leq n - 1$, clearing happens within the interval $\frac{\alpha_i^s}{\alpha_i^b} \theta^s \leq \theta^b \leq \frac{\alpha_{i+1}^s}{\alpha_{i+1}^b} \theta^s$ with a clearing price of $k\alpha_i^b \theta^b + (1-k)\alpha_i^s \theta^s$. Note that, we are assuming $\alpha_1^b \geq \alpha_2^b \geq \dots \geq \alpha_n^b$. For the utility calculation of the seller, the valuation is taken

Table 5.2: Clearing Conditions and Prices for Seller's Utility

#Units Clearing	Conditions	Clearing Price
1 Unit Clearing	$\alpha_1^s \theta^s \leq \alpha_1^b \theta^b$ and $\alpha_2^b \theta^b \leq \alpha_2^s \theta^s$ $\frac{\alpha_1^s}{\alpha_1^b} \theta^s \leq \theta^b \leq \frac{\alpha_2^s}{\alpha_2^b} \theta^s$	$k\alpha_1^b \theta^b + (1 - k)\alpha_1^s \theta^s$
i Unit Clearing	$\alpha_i^s \theta^s \leq \alpha_i^b \theta^b$ and $\alpha_{i+1}^b \theta^b \leq \alpha_{i+1}^s \theta^s$ $\frac{\alpha_i^s}{\alpha_i^b} \theta^s \leq \theta^b \leq \frac{\alpha_{i+1}^s}{\alpha_{i+1}^b} \theta^s$	$k\alpha_i^b \theta^b + (1 - k)\alpha_i^s \theta^s$
n Unit Clearing	$\alpha_n^s \theta^s \leq \alpha_n^b \theta^b$ and $\alpha_n^b \theta^b \leq \alpha_n^b h^b$ $\frac{\alpha_n^s}{\alpha_n^b} \theta^s \leq \theta^b \leq h^b$	$k\alpha_n^b \theta^b + (1 - k)\alpha_n^s \theta^s$

as negative as the seller has negative utility for the commodity, and the payment (which is a clearing price) is positive as the seller receives the amount.

Below, we start with the utility for the total clearing case u_n^s ,

$$u_n^s = n \int_{\frac{\alpha_n^s}{\alpha_n^b} \theta^s}^{h^b} \left((k\alpha_n^b \theta^b + (1 - k)\alpha_n^s \theta^s) - \theta^s \right) \cdot \left(\frac{1}{h^b - l^b} \right) d\theta^b \quad (5.7)$$

For $i \in \{1, \dots, n-1\}$, the utilities are as following,

$$u_i^s = i \int_{\frac{\alpha_i^s}{\alpha_i^b} \theta^s}^{\frac{\alpha_{i+1}^s}{\alpha_{i+1}^b} \theta^s} \left((k\alpha_i^b \theta^b + (1 - k)\alpha_i^s \theta^s) - \theta^s \right) \cdot \left(\frac{1}{h^b - l^b} \right) d\theta^b \quad (5.8)$$

Thus, the total utility U^s for the seller would be,

$$U^s = \int_{l^s}^{h^s} \sum_{i=1}^n u_i^s \cdot \left(\frac{1}{h^s - l^s} \right) d\theta^s \quad (5.9)$$

Equations 5.7 and 5.8 calculate the interim utility of the seller for the scenarios depending on number of units of clearance. The integral limits in equations 5.7 and 5.8 denote the regions where that many units of clearance happen. As we assume that the true types are sampled from the uniform distribution, the normalizing factors denote the probabilities of sampling from such a distribution. Equation 5.9 calculates the resultant expected utility of the seller. In order to solve Equation 5.9, we first need to solve equations 5.7 and 5.8. A simple integration and then simplification would lead to the following equations,

$$u_n^s = \left(\frac{n}{h^b - l^b} \right) \left[((1-k)\alpha_n^s - 1) \left(h^b - \frac{\alpha_n^s}{\alpha_n^b} \theta^s \right) \theta^s + \frac{k\alpha_n^b}{2} \left((h^b)^2 - \left(\frac{\alpha_n^s}{\alpha_n^b} \theta^s \right)^2 \right) \right] \quad (5.10)$$

For $i \in \{1, \dots, n-1\}$,

$$u_i^s = \left(\frac{i(\theta^s)^2}{h^b - l^b} \right) \left[((1-k)\alpha_i^s - 1) \left(\frac{\alpha_{i+1}^s}{\alpha_{i+1}^b} - \frac{\alpha_i^s}{\alpha_i^b} \right) + \frac{k\alpha_i^b}{2} \left(\left(\frac{\alpha_{i+1}^s}{\alpha_{i+1}^b} \right)^2 - \left(\frac{\alpha_i^s}{\alpha_i^b} \right)^2 \right) \right] \quad (5.11)$$

Now, as per Equation 5.9, we need to take a sum over all the individual interim utilities $u_i^s, 1 \leq i \leq n$ using equations 5.10 and 5.11.

$$\begin{aligned}
\sum_{j=1}^n u_j^s &= u_1^s + u_2^s + \dots + u_n^s \\
\sum_{j=1}^n u_j^s &= \left(\frac{(\theta^s)^2}{h^b - l^b} \right) \left[((1-k)\alpha_1^s - 1) \left(\frac{\alpha_2^s}{\alpha_2^b} - \frac{\alpha_1^s}{\alpha_1^b} \right) + \frac{k\alpha_1^b}{2} \left(\left(\frac{\alpha_2^s}{\alpha_2^b} \right)^2 - \left(\frac{\alpha_1^s}{\alpha_1^b} \right)^2 \right) \right] + \\
&\quad \left(\frac{2(\theta^s)^2}{h^b - l^b} \right) \left[((1-k)\alpha_2^s - 1) \left(\frac{\alpha_3^s}{\alpha_3^b} - \frac{\alpha_2^s}{\alpha_2^b} \right) + \frac{k\alpha_2^b}{2} \left(\left(\frac{\alpha_3^s}{\alpha_3^b} \right)^2 - \left(\frac{\alpha_2^s}{\alpha_2^b} \right)^2 \right) \right] + \dots + \\
&\quad \left(\frac{n}{h^b - l^b} \right) \left[((1-k)\alpha_n^s - 1) \left(h^b - \frac{\alpha_n^s}{\alpha_n^b} \theta^s \right) \theta^s + \frac{k\alpha_n^b}{2} \left((h^b)^2 - \left(\frac{\alpha_n^s}{\alpha_n^b} \theta^s \right)^2 \right) \right] \\
\sum_{j=1}^n u_j^s &= \left(\frac{1}{h^b - l^b} \right) \left[-(\theta^s)^2 \left(\frac{\alpha_2^s}{\alpha_2^b} - \frac{\alpha_1^s}{\alpha_1^b} \right) - 2(\theta^s)^2 \left(\frac{\alpha_3^s}{\alpha_3^b} - \frac{\alpha_2^s}{\alpha_2^b} \right) - \dots - n\theta^s \left(h^b - \frac{\alpha_n^s}{\alpha_n^b} \theta^s \right) + \right. \\
&\quad \left(1-k \right) \left(\alpha_1^s (\theta^s)^2 \left(\frac{\alpha_2^s}{\alpha_2^b} - \frac{\alpha_1^s}{\alpha_1^b} \right) + 2\alpha_2^s (\theta^s)^2 \left(\frac{\alpha_3^s}{\alpha_3^b} - \frac{\alpha_2^s}{\alpha_2^b} \right) + \dots + n\alpha_n^s \theta^s \left(h^b - \frac{\alpha_n^s}{\alpha_n^b} \theta^s \right) \right) + \frac{k}{2} \\
&\quad \left. \left((\theta^s)^2 \alpha_1^b \left(\left(\frac{\alpha_2^s}{\alpha_2^b} \right)^2 - \left(\frac{\alpha_1^s}{\alpha_1^b} \right)^2 \right) + 2(\theta^s)^2 \alpha_2^b \left(\left(\frac{\alpha_3^s}{\alpha_3^b} \right)^2 - \left(\frac{\alpha_2^s}{\alpha_2^b} \right)^2 \right) + \dots + n\alpha_n^b \left((h^b)^2 - \left(\frac{\alpha_n^s}{\alpha_n^b} \theta^b \right)^2 \right) \right) \right] \\
\sum_{j=1}^n u_j^s &= \left(\frac{1}{h^b - l^b} \right) \left[(\theta^s)^2 \left(\frac{\alpha_1^s}{\alpha_1^b} \right) + (\theta^s)^2 \left(\frac{\alpha_2^s}{\alpha_2^b} \right) + \dots + (\theta^s)^2 \left(\frac{\alpha_n^s}{\alpha_n^b} \right) - n\theta^s h^b \right. \\
&\quad \left. + \left\{ (\theta^s)^2 \left(\frac{\alpha_2^s}{\alpha_2^b} - \frac{\alpha_1^s}{\alpha_1^b} \right) \left((1-k)\alpha_1^s + \frac{k}{2}\alpha_1^b \left(\frac{\alpha_2^s}{\alpha_2^b} + \frac{\alpha_1^s}{\alpha_1^b} \right) \right) + 2(\theta^s)^2 \left(\frac{\alpha_3^s}{\alpha_3^b} - \frac{\alpha_2^s}{\alpha_2^b} \right) \right. \\
&\quad \left. \left. \left((1-k)\alpha_2^s + \frac{k}{2}\alpha_2^b \left(\frac{\alpha_3^s}{\alpha_3^b} + \frac{\alpha_2^s}{\alpha_2^b} \right) \right) + \dots + n \left(h^b - \frac{\alpha_n^s}{\alpha_n^b} \theta^s \right) \left((1-k)\theta^s \alpha_n^s + \frac{k}{2}\alpha_n^b \left(h^b + \frac{\alpha_n^s}{\alpha_n^b} \theta^s \right) \right) \right\} \right] \\
\sum_{j=1}^n u_j^s &= \frac{1}{h^b - l^b} \left[\sum_{j=1}^n \frac{\alpha_j^s}{\alpha_j^b} (\theta^s)^2 - n\theta^s h^b + \sum_{j=1}^{n-1} j(\theta^s)^2 \left(\frac{\alpha_{j+1}^s}{\alpha_{j+1}^b} - \frac{\alpha_j^s}{\alpha_j^b} \right) \left((1-k)\alpha_j^s + \frac{k\alpha_j^b}{2} \left(\frac{\alpha_{j+1}^s}{\alpha_{j+1}^b} + \frac{\alpha_j^s}{\alpha_j^b} \right) \right) \right. \\
&\quad \left. - \frac{n(1-k)}{\alpha_n^b} (\alpha_n^s)^2 (\theta^s)^2 - \frac{nk(\alpha_n^s)^2}{2\alpha_n^b} (\theta^s)^2 + \frac{nk}{2} \alpha_n^s \theta^s h^b + n(1-k)\alpha_n^s \theta^s h^b - \frac{nk}{2} \alpha_n^s \theta^s h^b \right. \\
&\quad \left. + \frac{nk}{2} \alpha_n^b (h^b)^2 \right] \\
\sum_{j=1}^n u_j^s &= \frac{1}{h^b - l^b} \left[\left(\sum_{j=1}^n \frac{\alpha_j^s}{\alpha_j^b} + \sum_{j=1}^{n-1} j \left(\frac{\alpha_{j+1}^s}{\alpha_{j+1}^b} - \frac{\alpha_j^s}{\alpha_j^b} \right) \left((1-k)\alpha_j^s + \frac{k\alpha_j^b}{2} \left(\frac{\alpha_{j+1}^s}{\alpha_{j+1}^b} + \frac{\alpha_j^s}{\alpha_j^b} \right) \right) \right. \right. \\
&\quad \left. \left. + \frac{n(k-2)}{2} \frac{(\alpha_n^s)^2}{\alpha_n^b} \right) (\theta^s)^2 + ((1-k)\alpha_n^s - 1) nh^b \theta^s + \frac{nk}{2(h^b - l^b)} \alpha_n^b (h^b)^2 \right]
\end{aligned}$$

Based on the value of $\sum_{j=1}^n u_j^s$, Solving Equation 5.9 and performing simplification would result in the below final utility of the seller,

$$U^s = \frac{(h^s)^2 + h^s l^s + (l^s)^2}{3(h^b - l^b)} \left(\sum_{j=1}^n \frac{\alpha_j^s}{\alpha_j^b} + \sum_{j=1}^{n-1} j \left(\frac{\alpha_{j+1}^s}{\alpha_{j+1}^b} - \frac{\alpha_j^s}{\alpha_j^b} \right) \left((1-k)\alpha_j^s + \frac{k\alpha_j^b}{2} \left(\frac{\alpha_{j+1}^s}{\alpha_{j+1}^b} + \frac{\alpha_j^s}{\alpha_j^b} \right) \right) \right. \\ \left. + \frac{n(k-2)}{2} \frac{(\alpha_n^s)^2}{\alpha_n^b} \right) + \frac{(h^s + l^s)}{2(h^b - l^b)} ((1-k)\alpha_n^s - 1) nh^b + \frac{nk}{2(h^b - l^b)} \alpha_n^b (h^b)^2 \quad (5.12)$$

5.3.1 Special case of $n = 1$ and $k = 0.5$

Below, we present the numerical analysis for the simplest case of $n = 1$ units of k -double auctions, where $k = 0.5$. For simplicity, let us assume that $\theta^b \sim U[0, 1]$ and $\theta^s \sim U[0, 1]$, which would make $l^b = l^s = 0$ and $h^b = h^s = 1$. As only 1 unit of commodity is traded, both buyer and seller have only a single scale factor, represented by α^b and α^s , respectively. Below, we rewrite equations 5.6 and 5.12 for this scenario,

$$U^b = \frac{1}{3} \left(\frac{\alpha^b}{\alpha^s} - \frac{3}{4} \frac{(\alpha^b)^2}{\alpha^s} \right) \\ U^s = \frac{1}{3} \left(\frac{\alpha^s}{\alpha^b} - \frac{3}{4} \frac{(\alpha^s)^2}{\alpha^b} \right) + \frac{1}{2} \left(\frac{\alpha^s}{2} - 1 \right) + \frac{\alpha^b}{4}$$

Now, to find the scale factor α^b that *maximize the utility* U^b for buyer, we take $\frac{\partial U^b}{\partial \alpha^b} = 0$, which implies,

$$\frac{1}{\alpha^s} - \frac{3}{2} \frac{\alpha^b}{\alpha^s} = 0 \Rightarrow \alpha^b = \frac{2}{3} \quad (5.13)$$

Similarly, we take $\frac{\partial U^s}{\partial \alpha^s} = 0$ for the seller, which implies,

$$\frac{1}{3} \left(\frac{1}{\alpha^b} - \frac{3}{2} \frac{\alpha^s}{\alpha^b} \right) + \frac{1}{4} = 0 \Rightarrow 2 - 3\alpha^s + \frac{3}{2}\alpha^b = 0 \Rightarrow \alpha^s = \frac{2}{3} + \frac{\alpha^b}{2} \quad (5.14)$$

As we know from Equation 5.13 that $\alpha^b = \frac{2}{3}$, putting the value of α^b in Equation 5.14 would result in $\alpha^s = 1$.

Remark For a specific case of k -double auction with $k = 0.5$, $\theta^b \sim U[0, 1]$ and $\theta^s \sim U[0, 1]$, where only a single unit of commodity is traded ($n = 1$), the scale-factors that lead to BNE are $\alpha^b = \frac{2}{3}$ and $\alpha^s = 1$.

5.3.2 Special case of $n = 2$ and $k = 0.5$

Below, we present the numerical analysis for the case of $n = 2$ units of k -double auctions, where $k = 0.5$. For simplicity, let us assume that $\theta^b \sim U[0, 1]$ and $\theta^s \sim U[0, 1]$, which would make $l^b = l^s = 0$ and $h^b = h^s = 1$. In such auctions, the buyer[sellers] places an individual bid[ask] for each of the two units of the commodity by following a scale-based bidding strategy. Below, we rewrite equations 5.6 and 5.12 for this scenario,

$$U^b = \frac{1}{3} \left(\frac{\alpha_1^b}{\alpha_1^s} + \frac{\alpha_2^b}{\alpha_2^s} - \left(\frac{\alpha_1^b}{\alpha_1^s} - \frac{\alpha_2^b}{\alpha_2^s} \right) \left(\frac{\alpha_1^b}{2} + \frac{\alpha_1^s}{4} \left(\frac{\alpha_1^b}{\alpha_1^s} + \frac{\alpha_2^b}{\alpha_2^s} \right) \right) - \frac{3}{2} \frac{(\alpha_2^b)^2}{\alpha_2^s} \right) \quad (5.15)$$

$$U^s = \frac{1}{3} \left(\frac{\alpha_1^s}{\alpha_1^b} + \frac{\alpha_2^s}{\alpha_2^b} + \left(\frac{\alpha_2^s}{\alpha_2^b} - \frac{\alpha_1^s}{\alpha_1^b} \right) \left(\frac{\alpha_1^s}{2} + \frac{\alpha_1^b}{4} \left(\frac{\alpha_2^s}{\alpha_2^b} + \frac{\alpha_1^s}{\alpha_1^b} \right) \right) - \frac{3}{2} \frac{(\alpha_2^s)^2}{\alpha_2^b} \right) + \frac{\alpha_2^s}{2} + \frac{\alpha_2^b}{2} - 1 \quad (5.16)$$

Here, the player may choose to place the same bid or ask by appropriately selecting the scale factors, leading to the following four possible cases.

Case-1: Same Scale Factors for Buyer, Same Scale Factors for Seller

Case-2: Different Scale Factors for Buyer, Same Scale Factors for Seller

Case-3: Same Scale Factors for Buyer, Different Scale Factors for Seller

Case-4: Different Scale-Factors for Buyer, Different Scale Factors for Seller

Below, we find the value of the scale factors for each of the four cases by utilising the analysis presented above.

Case-1: Same Scale Factors for Buyer, Same Scale Factors for Seller

In this case, the buyer [seller] places the same bids [asks] for both units of commodity; that is, the scale factors for the bids [asks] are the same; mathematically, $\alpha_1^b = \alpha_2^b = \alpha^b$ and $\alpha_1^s = \alpha_2^s = \alpha^s$. Replacing these values and rewriting equations 5.15 and 5.16,

$$\begin{aligned} U^b &= \frac{1}{3} \left(\frac{\alpha^b}{\alpha^s} + \frac{\alpha^b}{\alpha^s} - \left(\frac{\alpha^b}{\alpha^s} - \frac{\alpha^b}{\alpha^s} \right) \left(\frac{\alpha^b}{2} + \frac{\alpha^s}{4} \left(\frac{\alpha^b}{\alpha^s} + \frac{\alpha^b}{\alpha^s} \right) \right) - \frac{3}{2} \frac{(\alpha^b)^2}{\alpha^s} \right) \\ &= \frac{1}{3} \left(2 \frac{\alpha^b}{\alpha^s} - \frac{3}{2} \frac{(\alpha^b)^2}{\alpha^s} \right) \end{aligned}$$

$$\begin{aligned} U^s &= \frac{1}{3} \left(\frac{\alpha^s}{\alpha^b} + \frac{\alpha^s}{\alpha^b} + \left(\frac{\alpha^s}{\alpha^b} - \frac{\alpha^s}{\alpha^b} \right) \left(\frac{\alpha^s}{2} + \frac{\alpha^b}{4} \left(\frac{\alpha^s}{\alpha^b} + \frac{\alpha^s}{\alpha^b} \right) \right) - \frac{3}{2} \frac{(\alpha^s)^2}{\alpha^b} \right) + \frac{\alpha^s}{2} + \frac{\alpha^b}{2} - 1 \\ &= \frac{1}{3} \left(2 \frac{\alpha^s}{\alpha^b} - \frac{3}{2} \frac{(\alpha^s)^2}{\alpha^b} \right) + \frac{\alpha^s}{2} + \frac{\alpha^b}{2} - 1 \end{aligned}$$

Similar to previous scenario, to find the scale factor α^b that maximize the utility U^b for buyer, we take $\frac{\partial U^b}{\partial \alpha^b} = 0$, which implies,

$$\frac{2}{\alpha^s} - 3 \frac{\alpha^b}{\alpha^s} = 0 \Rightarrow \alpha^b = \frac{2}{3} \quad (5.17)$$

Similarly, we take $\frac{\partial U^s}{\partial \alpha^s} = 0$ for the seller, which implies,

$$\frac{1}{3} \left(\frac{2}{\alpha^b} - 3 \frac{\alpha^s}{\alpha^b} \right) + \frac{1}{2} = 0 \Rightarrow 2 - 3\alpha^s + \frac{3}{2}\alpha^b = 0 \Rightarrow \alpha^s = \frac{2}{3} + \frac{\alpha^b}{2} \quad (5.18)$$

As we know from Equation 5.17 that $\alpha^b = \frac{2}{3}$, putting the value of α^b in Equation 5.18 would result in $\alpha^s = 1$.

Remark For a specific case of k -double auction with $k = 0.5$, $\theta^b \sim U[0, 1]$ and $\theta^s \sim U[0, 1]$, where only two units of commodity are traded ($n = 2$) with $\alpha_1^b = \alpha_2^b = \alpha^b$ and $\alpha_1^s = \alpha_2^s = \alpha^s$, the scale-factors that lead to BNE are $\alpha^b = \frac{2}{3}$ and $\alpha^s = 1$.

Case-2: Different Scale Factors for Buyer, Same Scale Factors for Seller

In this case, only the seller places the same asks for both units of commodity, while the buyer places two different bids for the two units of commodity; that is, the scale factors for the asks are the same and different for bids; mathematically, $\alpha_1^s = \alpha_2^s = \alpha^s$ and $\alpha_1^b \neq \alpha_2^b$. Replacing these values and rewriting equations 5.15 and 5.16,

$$\begin{aligned} U^b &= \frac{1}{3} \left(\frac{\alpha_1^b}{\alpha^s} + \frac{\alpha_2^b}{\alpha^s} - \left(\frac{\alpha_1^b}{\alpha^s} - \frac{\alpha_2^b}{\alpha^s} \right) \left(\frac{\alpha_1^b}{2} + \frac{1}{4} \left(\frac{\alpha_1^b}{\alpha^s} + \frac{\alpha_2^b}{\alpha^s} \right) \right) \right) - \frac{3}{2} \frac{(\alpha_2^b)^2}{\alpha^s} \\ &= \frac{1}{3\alpha^s} \left(\alpha_1^b + \alpha_2^b - (\alpha_1^b - \alpha_2^b) \left(\frac{\alpha_1^b}{2} + \frac{1}{4} (\alpha_1^b + \alpha_2^b) \right) \right) - \frac{3}{2} (\alpha_2^b)^2 \\ U^s &= \frac{1}{3} \left(\frac{\alpha^s}{\alpha_1^b} + \frac{\alpha^s}{\alpha_2^b} + \left(\frac{\alpha^s}{\alpha_2^b} - \frac{\alpha^s}{\alpha_1^b} \right) \left(\frac{\alpha^s}{2} + \frac{\alpha_1^b}{4} \left(\frac{\alpha^s}{\alpha_2^b} + \frac{\alpha^s}{\alpha_1^b} \right) \right) \right) - \frac{3}{2} \frac{(\alpha^s)^2}{\alpha_2^b} + \frac{\alpha^s}{2} + \frac{\alpha_2^b}{2} - 1 \end{aligned}$$

To find the scale factor α_1^b and α_2^b that *maximize the utility* U^b for buyer, we take $\frac{\partial U^b}{\partial \alpha_1^b} = 0$ and $\frac{\partial U^b}{\partial \alpha_2^b} = 0$, which implies,

$$\begin{aligned} \frac{\partial U^b}{\partial \alpha_1^b} = 0 &\Rightarrow \left[1 - (\alpha_1^b - \alpha_2^b) \frac{3}{4} - \left(\frac{\alpha_1^b}{2} + \frac{1}{4} (\alpha_1^b + \alpha_2^b) \right) \right] = 0 \\ &\Rightarrow 1 - \frac{3}{2} \alpha_1^b + \frac{1}{2} \alpha_2^b \Rightarrow 3\alpha_1^b - \alpha_2^b = 2 \end{aligned} \tag{5.19}$$

$$\begin{aligned} \frac{\partial U^b}{\partial \alpha_2^b} = 0 &\Rightarrow \left[1 - (\alpha_1^b - \alpha_2^b) \frac{1}{4} + \left(\frac{\alpha_1^b}{2} + \frac{1}{4} (\alpha_1^b + \alpha_2^b) \right) - 3\alpha_2^b \right] = 0 \\ &\Rightarrow 1 + \frac{1}{2} \alpha_1^b - \frac{5}{2} \alpha_2^b = 0 \Rightarrow \alpha_1^b - 5\alpha_2^b = -2 \end{aligned} \tag{5.20}$$

Similarly, we take $\frac{\partial U^s}{\partial \alpha^s} = 0$ for the seller, which implies,

$$\begin{aligned}
& \frac{1}{3} \left(\frac{1}{\alpha_1^b} + \frac{1}{\alpha_2^b} + \left(\frac{1}{\alpha_2^b} - \frac{1}{\alpha_1^b} \right) \left(\frac{\alpha^s}{2} + \frac{\alpha_1^b}{4} \left(\frac{\alpha^s}{\alpha_2^b} + \frac{\alpha^s}{\alpha_1^b} \right) \right) + \left(\frac{\alpha^s}{\alpha_2^b} - \frac{\alpha^s}{\alpha_1^b} \right) \left(\frac{1}{2} + \frac{\alpha_1^b}{4} \left(\frac{1}{\alpha_2^b} + \frac{1}{\alpha_1^b} \right) \right) \right. \\
& \left. - \frac{3\alpha_s}{\alpha_2^b} \right) + \frac{1}{2} = 0 \\
& \frac{1}{3} \left(\frac{1}{\alpha_1^b} + \frac{1}{\alpha_2^b} + \alpha^s \left(\frac{1}{\alpha_2^b} - \frac{1}{\alpha_1^b} \right) \left(1 + \frac{\alpha_1^b}{2} \left(\frac{1}{\alpha_2^b} + \frac{1}{\alpha_1^b} \right) \right) \right) - \frac{\alpha_s}{\alpha_2^b} + \frac{1}{2} = 0
\end{aligned} \tag{5.21}$$

By solving Equation 5.19 and Equation 5.20, we get $\alpha_1^b = \frac{6}{7}$ and $\alpha_2^b = \frac{4}{7}$. Then, substituting the values of α_1^b and α_2^b in Equation 5.21 would result in $\alpha^s = \frac{212}{189}$.

Remark For a specific case of k -double auction with $k = 0.5$, $\theta^b \sim U[0, 1]$ and $\theta^s \sim U[0, 1]$, where only two units of commodity are traded ($n = 2$) with $\alpha_1^b \neq \alpha_2^b$ and $\alpha_1^s = \alpha_2^s = \alpha^s$, the scale-factors that lead to BNE are $\alpha_1^b = \frac{6}{7}$, $\alpha_2^b = \frac{4}{7}$ and $\alpha^s = \frac{212}{189}$.

Case-3: Same Scale Factors for Buyer, Different Scale Factors for Seller

In this case, only the buyer places the same bids for both units of commodity, while the seller places two different asks for the two units of commodity; that is, the scale factors for the bids are the same and different for asks; mathematically, $\alpha_1^b = \alpha_2^b = \alpha^b$ and $\alpha_1^s \neq \alpha_2^s$. Replacing these values and rewriting equations 5.15 and 5.16,

$$\begin{aligned}
U^b &= \frac{1}{3} \left(\frac{\alpha^b}{\alpha_1^s} + \frac{\alpha^b}{\alpha_2^s} - \left(\frac{\alpha^b}{\alpha_1^s} - \frac{\alpha^b}{\alpha_2^s} \right) \left(\frac{\alpha^b}{2} + \frac{\alpha_1^s}{4} \left(\frac{\alpha^b}{\alpha_1^s} + \frac{\alpha^b}{\alpha_2^s} \right) \right) - \frac{3}{2} \frac{(\alpha^b)^2}{\alpha_2^s} \right) \\
U^s &= \frac{1}{3} \left(\frac{\alpha_1^s}{\alpha^b} + \frac{\alpha_2^s}{\alpha^b} + \left(\frac{\alpha_2^s}{\alpha^b} - \frac{\alpha_1^s}{\alpha^b} \right) \left(\frac{\alpha_1^s}{2} + \frac{\alpha^b}{4} \left(\frac{\alpha_2^s}{\alpha^b} + \frac{\alpha_1^s}{\alpha^b} \right) \right) - \frac{3}{2} \frac{(\alpha_2^s)^2}{\alpha^b} \right) + \frac{\alpha_2^s}{2} + \frac{\alpha^b}{2} - 1 \\
&= \frac{1}{3\alpha^b} \left(\alpha_1^s + \alpha_2^s + (\alpha_2^s - \alpha_1^s) \left(\frac{\alpha_1^s}{2} + \frac{1}{4} (\alpha_2^s + \alpha_1^s) \right) - \frac{3}{2} (\alpha_2^s)^2 \right) + \frac{\alpha_2^s}{2} + \frac{\alpha^b}{2} - 1
\end{aligned}$$

To find the scale factor α_1^s and α_2^s that **maximize the utility** U^s for seller, we take $\frac{\partial U^s}{\partial \alpha_1^s} = 0$ and $\frac{\partial U^s}{\partial \alpha_2^s} = 0$, which implies,

$$\begin{aligned}\frac{\partial U^s}{\partial \alpha_1^s} = 0 &\Rightarrow \left[1 + (\alpha_2^s - \alpha_1^s) \left(\frac{1}{2} + \frac{1}{4} \right) - \left(\frac{\alpha_1^s}{2} + \frac{1}{4}(\alpha_2^s + \alpha_1^s) \right) \right] = 0 \\ &\Rightarrow 1 - \frac{3}{2}\alpha_1^s + \frac{1}{2}\alpha_2^s = 0 \Rightarrow 3\alpha_1^s - \alpha_2^s = 2\end{aligned}\tag{5.22}$$

$$\begin{aligned}\frac{\partial U^s}{\partial \alpha_2^s} = 0 &\Rightarrow \frac{1}{3\alpha^b} \left[1 + (\alpha_2^s - \alpha_1^s) \frac{1}{4} + \left(\frac{\alpha_1^s}{2} + \frac{1}{4}(\alpha_2^s + \alpha_1^s) \right) - 3\alpha_2^s \right] + \frac{1}{2} = 0 \\ &\Rightarrow \frac{1}{3\alpha^b} \left[1 - \frac{5}{2}\alpha_2^s + \frac{1}{2}\alpha_1^s \right] + \frac{1}{2} = 0 \Rightarrow \alpha_1^s - 5\alpha_2^s + 3\alpha^b = -2\end{aligned}\tag{5.23}$$

Similarly, we take $\frac{\partial U^b}{\partial \alpha^b} = 0$ for the seller, which implies,

$$\begin{aligned}&\frac{1}{\alpha_1^s} + \frac{1}{\alpha_2^s} - \left(\frac{1}{\alpha_1^s} - \frac{1}{\alpha_2^s} \right) \left(\frac{\alpha^b}{2} + \frac{\alpha_1^s}{4} \left(\frac{\alpha^b}{\alpha_1^s} + \frac{\alpha^b}{\alpha_2^s} \right) \right) - \left(\frac{\alpha^b}{\alpha_1^s} - \frac{\alpha^b}{\alpha_2^s} \right) \left(\frac{1}{2} + \frac{\alpha_1^s}{4} \left(\frac{1}{\alpha_1^s} + \frac{1}{\alpha_2^s} \right) \right) \\ &- \frac{3\alpha^b}{\alpha_2^s} = 0 \\ &\frac{1}{\alpha_1^s} + \frac{1}{\alpha_2^s} - \alpha^b \left(\frac{1}{\alpha_1^s} - \frac{1}{\alpha_2^s} \right) \left(\frac{1}{2} + \frac{\alpha_1^s}{4} \left(\frac{1}{\alpha_1^s} + \frac{1}{\alpha_2^s} \right) + \frac{1}{2} + \frac{\alpha_1^s}{4} \left(\frac{1}{\alpha_1^s} + \frac{1}{\alpha_2^s} \right) \right) - \frac{3\alpha^b}{\alpha_2^s} = 0 \\ &\left(\frac{1}{\alpha_1^s} + \frac{1}{\alpha_2^s} \right) - \alpha^b \left(\frac{1}{\alpha_1^s} - \frac{1}{\alpha_2^s} \right) \left(1 + \frac{\alpha_1^s}{2} \left(\frac{1}{\alpha_1^s} + \frac{1}{\alpha_2^s} \right) \right) - \frac{3\alpha^b}{\alpha_2^s} = 0\end{aligned}\tag{5.24}$$

By solving Equation 5.22, Equation 5.23 and Equation 5.24, we get $\alpha_1^s = 1$ and $\alpha_2^s = 1$ and $\alpha^b = \frac{2}{3}$.

Remark For a specific case of k -double auction with $k = 0.5$, $\theta^b \sim U[0, 1]$ and $\theta^s \sim U[0, 1]$, where only two units of commodity are traded ($n = 2$) with $\alpha_1^s \neq \alpha_2^s$ and $\alpha_1^b = \alpha_2^b = \alpha^b$, the scale-factors that lead to BNE are $\alpha^b = \frac{2}{3}$, $\alpha_1^s = 1$ and $\alpha_2^s = 1$.

Here, even though we started with the condition that $\alpha_1^s \neq \alpha_2^s$, the BNE occurs when $\alpha_1^s = \alpha_2^s$. This case is particularly similar to **Case-1**.

Case-4: Different Scale Factors for Buyer, Different Scale Factors for Seller

In this case, the buyer places two different bids, and also the seller places two different asks for the two units of commodity; that is, the scale factors for both the bids and the asks are different; mathematically, $\alpha_1^b \neq \alpha_2^b$ and $\alpha_1^s \neq \alpha_2^s$. Refer to the equations 5.15 and 5.16,

$$U^b = \frac{1}{3} \left(\frac{\alpha_1^b}{\alpha_1^s} + \frac{\alpha_2^b}{\alpha_2^s} - \left(\frac{\alpha_1^b}{\alpha_1^s} - \frac{\alpha_2^b}{\alpha_2^s} \right) \left(\frac{\alpha_1^b}{2} + \frac{1}{4} \left(\frac{\alpha_1^b}{\alpha_1^s} + \frac{\alpha_2^b}{\alpha_2^s} \right) \right) - \frac{3}{2} \frac{(\alpha_2^b)^2}{\alpha_2^s} \right) \quad (5.25)$$

$$U^s = \frac{1}{3} \left(\frac{\alpha_1^s}{\alpha_1^b} + \frac{\alpha_2^s}{\alpha_2^b} + \left(\frac{\alpha_2^s}{\alpha_2^b} - \frac{\alpha_1^s}{\alpha_1^b} \right) \left(\frac{\alpha_2^s}{2} + \frac{1}{4} \left(\frac{\alpha_2^s}{\alpha_2^b} + \frac{\alpha_1^s}{\alpha_1^b} \right) \right) - \frac{3}{2} \frac{(\alpha_2^s)^2}{\alpha_2^b} \right) + \frac{\alpha_2^s}{2} + \frac{\alpha_2^b}{2} - 1 \quad (5.26)$$

To find the scale factor α_1^b and α_2^b that **maximize the utility** U^b for buyer, we take $\frac{\partial U^b}{\partial \alpha_1^b} = 0$ and $\frac{\partial U^b}{\partial \alpha_2^b} = 0$, which implies,

$$\begin{aligned} \frac{\partial U^b}{\partial \alpha_1^b} = 0 &\Rightarrow \frac{1}{\alpha_1^s} - \left(\frac{\alpha_1^b}{\alpha_1^s} - \frac{\alpha_2^b}{\alpha_2^s} \right) \left(\frac{1}{2} + \frac{1}{4} \right) - \frac{1}{\alpha_1^s} \left(\frac{\alpha_1^b}{2} + \frac{1}{4} \left(\frac{\alpha_1^b}{\alpha_1^s} + \frac{\alpha_2^b}{\alpha_2^s} \right) \right) = 0 \\ &\Rightarrow \alpha_2^s - \frac{3}{4} (\alpha_1^b \alpha_2^s - \alpha_2^b \alpha_1^s) - \frac{1}{2} \alpha_1^b \alpha_2^s - \frac{1}{4} (\alpha_1^b \alpha_2^s + \alpha_2^b \alpha_1^s) = 0 \\ &\Rightarrow \alpha_2^s - \alpha_1^b \alpha_2^s + \frac{1}{2} \alpha_2^b \alpha_1^s - \frac{1}{2} \alpha_1^b \alpha_2^s = 0 \\ &\Rightarrow 2\alpha_2^s - 3\alpha_1^b \alpha_2^s + \alpha_2^b \alpha_1^s = 0 \end{aligned} \quad (5.27)$$

$$\begin{aligned} \frac{\partial U^b}{\partial \alpha_2^b} = 0 &\Rightarrow \frac{1}{\alpha_2^s} - \left(\frac{\alpha_1^b}{\alpha_1^s} - \frac{\alpha_2^b}{\alpha_2^s} \right) \left(\frac{\alpha_1^s}{4\alpha_2^s} \right) + \frac{1}{\alpha_2^s} \left(\frac{\alpha_1^b}{2} + \frac{1}{4} \left(\frac{\alpha_1^b}{\alpha_1^s} + \frac{\alpha_2^b}{\alpha_2^s} \right) \right) - \frac{3\alpha_2^b}{\alpha_2^s} = 0 \\ &\Rightarrow 1 - \frac{\alpha_1^b}{4} + \frac{\alpha_2^b \alpha_1^s}{4\alpha_2^s} + \frac{\alpha_1^b}{2} + \frac{\alpha_2^b}{4} + \frac{\alpha_1^s \alpha_2^b}{4\alpha_2^s} - 3\alpha_2^b = 0 \\ &\Rightarrow 4\alpha_2^s + 2\alpha_1^b \alpha_2^s + 2\alpha_2^b \alpha_1^s - 12\alpha_2^b \alpha_2^s = 0 \\ &\Rightarrow \alpha_1^s \alpha_2^b + \alpha_2^s (\alpha_1^b - 6\alpha_2^b + 2) = 0 \end{aligned} \quad (5.28)$$

Similarly, we take $\frac{\partial U^s}{\partial \alpha_1^s} = 0$ and $\frac{\partial U^s}{\partial \alpha_2^s} = 0$ for the seller, which implies,

$$\begin{aligned}
\frac{\partial U^s}{\partial \alpha_1^s} = 0 &\Rightarrow \frac{1}{\alpha_1^b} + \left(\frac{\alpha_2^s}{\alpha_2^b} - \frac{\alpha_1^s}{\alpha_1^b} \right) \left(\frac{1}{2} + \frac{1}{4} \right) - \left(\frac{1}{\alpha_1^b} \right) \left(\frac{\alpha_1^s}{2} + \frac{\alpha_1^b}{4} \left(\frac{\alpha_2^s}{\alpha_2^b} + \frac{\alpha_1^s}{\alpha_1^b} \right) \right) = 0 \\
&\Rightarrow 4\alpha_2^b + 3\alpha_1^b\alpha_2^s - 3\alpha_1^s\alpha_2^b - 2\alpha_1^s\alpha_2^b - \alpha_2^s\alpha_1^b - \alpha_1^s\alpha_2^b = 0 \\
&\Rightarrow 2\alpha_2^b - 3\alpha_1^s\alpha_2^b + \alpha_2^s\alpha_1^b = 0
\end{aligned} \tag{5.29}$$

$$\begin{aligned}
\frac{\partial U^s}{\partial \alpha_2^s} = 0 &\Rightarrow \frac{1}{3} \left[\frac{1}{\alpha_2^b} + \left(\frac{\alpha_2^s}{\alpha_2^b} - \frac{\alpha_1^s}{\alpha_1^b} \right) \left(\frac{\alpha_1^b}{4\alpha_2^b} \right) + \left(\frac{1}{\alpha_2^b} \right) \left(\frac{\alpha_1^s}{2} + \frac{\alpha_1^b}{4} \left(\frac{\alpha_2^s}{\alpha_2^b} + \frac{\alpha_1^s}{\alpha_1^b} \right) \right) - \frac{3\alpha_2^s}{\alpha_2^b} \right] + \frac{1}{2} = 0 \\
&\Rightarrow \frac{1}{3} \left[\frac{1}{\alpha_2^b} + \left(\frac{\alpha_2^s\alpha_1^b}{4(\alpha_2^b)^2} - \frac{\alpha_1^s}{4\alpha_2^b} \right) + \left(\frac{\alpha_1^s}{2\alpha_2^b} + \frac{\alpha_2^s\alpha_1^b}{4(\alpha_2^b)^2} + \frac{\alpha_1^s}{4\alpha_2^b} \right) - \frac{3\alpha_2^s}{\alpha_2^b} \right] + \frac{1}{2} = 0 \\
&\Rightarrow \frac{1}{3\alpha_2^b} + \frac{1}{6} \frac{\alpha_1^b\alpha_2^s}{(\alpha_2^b)^2} + \frac{\alpha_1^s}{6\alpha_2^b} - \frac{\alpha_2^s}{\alpha_2^b} + \frac{1}{2} = 0 \\
&\Rightarrow 2\alpha_2^b + \alpha_1^b\alpha_2^s + \alpha_1^s\alpha_2^b - 6\alpha_2^s\alpha_2^b + 3(\alpha_2^b)^2 = 0 \\
&\Rightarrow \alpha_1^b\alpha_2^s + \alpha_2^b(\alpha_1^s - 6\alpha_2^s + 2) + 3(\alpha_2^b)^2 = 0
\end{aligned} \tag{5.30}$$

By solving Equation 5.27, Equation 5.28, Equation 5.29 and Equation 5.30, we get $\alpha_1^b \approx 0.882782$ and $\alpha_2^b \approx 0.588521$, $\alpha_1^s \approx 1.2207$ and $\alpha_2^s \approx 1.10806$.

Remark For a specific case of k -double auction with $k = 0.5$, $\theta^b \sim U[0, 1]$ and $\theta^s \sim U[0, 1]$, where only two units of commodity are traded ($n = 2$) with $\alpha_1^b \neq \alpha_2^b$ and $\alpha_1^s \neq \alpha_2^s$, the scale-factors that lead to BNE are $\alpha_1^b \approx 0.882782$ and $\alpha_2^b \approx 0.588521$, $\alpha_1^s \approx 1.2207$ and $\alpha_2^s \approx 1.10806$.

Theorem 5.1. *For a single-buyer single-seller two-unit k -double auction with $k = 0.5$; $\theta^b \sim U[0, 1]$ and $\theta^s \sim U[0, 1]$, respectively, when they deploy scale based bidding strategies b_B and b_S and fix their scaling factors before seeing their true types, scale factors shown in Table 5.3 constitute the BNE for each case.*

Table 5.3: Value of Scale Factors for Various Cases

Case-1: $\alpha_1^b = \alpha_2^b = \alpha^b$ and $\alpha_1^s = \alpha_2^s = \alpha^s$	$\alpha^b = \frac{2}{3}, \alpha^s = 1$
Case-2: $\alpha_1^b \neq \alpha_2^b$ and $\alpha_1^s = \alpha_2^s = \alpha^s$	$\alpha_1^b = \frac{6}{7}, \alpha_2^b = \frac{4}{7}, \alpha^s = \frac{212}{189}$
Case-3: $\alpha_1^b = \alpha_2^b = \alpha^b$ and $\alpha_1^s \neq \alpha_2^s$	$\alpha^b = \frac{2}{3}, \alpha_1^s = 1, \alpha_2^s = 1$
Case-4: $\alpha_1^b \neq \alpha_2^b$ and $\alpha_1^s \neq \alpha_2^s$	$\alpha_1^b \approx 0.882782, \alpha_2^b \approx 0.588521,$ $\alpha_1^s \approx 1.2207, \alpha_2^s \approx 1.10806$

5.3.3 Generalizing the Scale factors Calculation

The above analysis helps us to calculate the utilities of the buyer and seller for each specific case, subsequently leading to scale factors that maximize the calculated utility. Below, we provide the *generic equations* for all the scale factors; that is, we proceed to find the scale factors α_i^b that maximize the buyer's expected utility U^b defined in Equation 5.6 and the scale factors α_i^s that maximizes the seller's expected utility U^s defined in Equation 5.12. To achieve this, we take $\frac{\partial U^b}{\partial \alpha_i^b} = 0, \forall i \in \{1, 2, \dots, n\}$. Similarly, we take $\frac{\partial U^s}{\partial \alpha_i^s} = 0, \forall i \in \{1, 2, \dots, n\}$.

if $j = 1$, equating $\frac{\partial U^b}{\partial \alpha_1^b} = 0$ and $\frac{\partial U^s}{\partial \alpha_1^s} = 0$ would lead to following equations,

$$\begin{aligned}
 \frac{\partial U^b}{\partial \alpha_1^b} = 0 &\Rightarrow \frac{1}{\alpha_1^s} - \left(\frac{\alpha_1^b}{\alpha_1^s} - \frac{\alpha_2^b}{\alpha_2^s} \right) \left(k + \frac{1-k}{2} \right) - \frac{1}{\alpha_1^s} \left(k \alpha_1^b + \frac{(1-k)\alpha_1^s}{2} \left(\frac{\alpha_1^b}{\alpha_1^s} + \frac{\alpha_2^b}{\alpha_2^s} \right) \right) = 0 \\
 &\Rightarrow \frac{1}{\alpha_1^s} - \frac{(k+1)}{2} \left(\frac{\alpha_1^b}{\alpha_1^s} - \frac{\alpha_2^b}{\alpha_2^s} \right) - k \frac{\alpha_1^b}{\alpha_1^s} - \frac{(1-k)}{2} \left(\frac{\alpha_1^b}{\alpha_1^s} + \frac{\alpha_2^b}{\alpha_2^s} \right) = 0 \\
 &\Rightarrow \frac{1}{\alpha_1^s} - (k+1) \frac{\alpha_1^b}{\alpha_1^s} + k \frac{\alpha_2^b}{\alpha_2^s} = 0 \\
 &\Rightarrow (1+k) \alpha_1^b \alpha_2^s - k \alpha_2^b \alpha_1^s = \alpha_2^s
 \end{aligned}$$

$$\begin{aligned}
\frac{\partial U^s}{\partial \alpha_1^s} = 0 &\Rightarrow \frac{1}{\alpha_1^b} + \left(\frac{\alpha_2^s}{\alpha_2^b} - \frac{\alpha_1^s}{\alpha_1^b} \right) \left(1 - k + \frac{k}{2} \right) - \frac{1}{\alpha_1^b} \left((1-k)\alpha_1^s + \frac{k\alpha_1^b}{2} \left(\frac{\alpha_2^s}{\alpha_2^b} + \frac{\alpha_1^s}{\alpha_1^b} \right) \right) = 0 \\
&\Rightarrow \frac{1}{\alpha_1^b} + \frac{(2-k)}{2} \left(\frac{\alpha_2^s}{\alpha_2^b} - \frac{\alpha_1^s}{\alpha_1^b} \right) - (1-k) \frac{\alpha_1^s}{\alpha_1^b} - \frac{k}{2} \left(\frac{\alpha_2^s}{\alpha_2^b} + \frac{\alpha_1^s}{\alpha_1^b} \right) = 0 \\
&\Rightarrow \frac{1}{\alpha_1^b} + (1-k) \frac{\alpha_2^s}{\alpha_2^b} - (2-k) \frac{\alpha_1^s}{\alpha_1^b} = 0 \Rightarrow (1-k)\alpha_1^b \alpha_2^s - (2-k)\alpha_1^s \alpha_2^b = -\alpha_2^b
\end{aligned}$$

Thus, for $j = 1$, equating $\frac{\partial U^b}{\partial \alpha_1^b} = 0$ and $\frac{\partial U^s}{\partial \alpha_1^s} = 0$ would lead to following equations,

$$(1+k)\alpha_1^b \alpha_2^s - k\alpha_2^b \alpha_1^s = \alpha_2^s \quad (5.31)$$

$$(1-k)\alpha_1^b \alpha_2^s - (2-k)\alpha_1^s \alpha_2^b = -\alpha_2^b \quad (5.32)$$

Then, if $j = n$, equating $\frac{\partial U^b}{\partial \alpha_n^b} = 0$ and $\frac{\partial U^s}{\partial \alpha_n^s} = 0$ would lead to following equations,

$$\begin{aligned}
\frac{\partial U^b}{\partial \alpha_n^b} = 0 &\Rightarrow \frac{(h^b)^2 + h^b l^b + (l^b)^2}{3} \left[\frac{1}{\alpha_n^s} - (n-1) \left(\frac{\alpha_{n-1}^b}{\alpha_{n-1}^s} - \frac{\alpha_n^b}{\alpha_n^s} \right) \left(\frac{(1-k)}{2} \frac{\alpha_{n-1}^s}{\alpha_n^s} \right) + \right. \\
&\quad \left. \frac{n-1}{\alpha_n^s} \left(k\alpha_{n-1}^b + \frac{(1-k)\alpha_{n-1}^s}{2} \left(\frac{\alpha_{n-1}^b}{\alpha_{n-1}^s} + \frac{\alpha_n^b}{\alpha_n^s} \right) \right) - n(1+k) \frac{\alpha_n^b}{\alpha_n^s} \right] + \frac{(h^b + l^b)}{2} n k l^s = 0 \\
&\Rightarrow \frac{2((h^b)^2 + h^b l^b + (l^b)^2)}{h^b + l^b} \left[\frac{1}{\alpha_n^s} - \frac{(n-1)(1-k)}{2} \left(\frac{\alpha_{n-1}^b}{\alpha_n^s} - \frac{\alpha_n^b \alpha_{n-1}^s}{(\alpha_n^s)^2} \right) + \right. \\
&\quad \left. k \frac{(n-1)\alpha_{n-1}^b}{\alpha_n^s} + \frac{(n-1)(1-k)}{2\alpha_n^s} \left(\alpha_{n-1}^b + \frac{\alpha_n^b \alpha_{n-1}^s}{\alpha_n^s} \right) - n(1+k) \frac{\alpha_n^b}{\alpha_n^s} \right] + 3n k l^s = 0 \\
&\Rightarrow 1 - \frac{(n-1)(1-k)}{2} \left(\alpha_{n-1}^b - \frac{\alpha_n^b \alpha_{n-1}^s}{\alpha_n^s} \right) + k(n-1)\alpha_{n-1}^b - n(1+k)\alpha_n^b + \\
&\quad \frac{(n-1)(1-k)}{2} \left(\alpha_{n-1}^b + \frac{\alpha_n^b \alpha_{n-1}^s}{\alpha_n^s} \right) = -\frac{3(h^b + l^b) k n l^s \alpha_n^s}{2((h^b)^2 + h^b l^b + (l^b)^2)} \\
&\Rightarrow \frac{3(h^b + l^b) k n l^s \alpha_n^s}{2((h^b)^2 + h^b l^b + (l^b)^2)} + (n-1) \left((1-k) \frac{\alpha_n^b \alpha_{n-1}^s}{\alpha_n^s} + k \alpha_{n-1}^b \right) - n(k+1)\alpha_n^b + 1 = 0
\end{aligned}$$

$$\begin{aligned}
\frac{\partial U^s}{\partial \alpha_n^s} = 0 \Rightarrow & \frac{(h^s)^2 + h^s l^s + (l^s)^2}{3} \left[\frac{1}{\alpha_n^b} + (n-1) \left(\frac{\alpha_n^s}{\alpha_n^b} - \frac{\alpha_{n-1}^s}{\alpha_{n-1}^b} \right) \left(\frac{k}{2} \frac{\alpha_{n-1}^b}{\alpha_n^b} \right) + \right. \\
& \left. \frac{n-1}{\alpha_n^b} \left((1-k)\alpha_{n-1}^s + \frac{k\alpha_{n-1}^b}{2} \left(\frac{\alpha_n^s}{\alpha_n^b} + \frac{\alpha_{n-1}^s}{\alpha_{n-1}^b} \right) \right) + n(k-2) \frac{\alpha_n^s}{\alpha_n^b} \right] + \frac{(h^s + l^s)}{2} n(1-k) h^b = 0 \\
\Rightarrow & \frac{2((h^s)^2 + h^s l^s + (l^s)^2)}{h^s + l^s} \left[\frac{1}{\alpha_n^b} + \frac{k(n-1)}{2} \left(\frac{\alpha_n^s \alpha_{n-1}^b}{(\alpha_n^b)^2} - \frac{\alpha_{n-1}^s}{\alpha_n^b} \right) + \right. \\
& \left. (1-k)(n-1) \frac{\alpha_{n-1}^s}{\alpha_n^b} + \frac{k(n-1)}{2\alpha_n^b} \left(\frac{\alpha_n^s \alpha_{n-1}^b}{\alpha_n^b} + \alpha_{n-1}^s \right) + n(k-2) \frac{\alpha_n^s}{\alpha_n^b} \right] + 3n(1-k) h^b = 0 \\
\Rightarrow & 1 + \frac{k(n-1)}{2} \left(\frac{\alpha_n^s \alpha_{n-1}^b}{\alpha_n^b} - \alpha_{n-1}^s \right) + (1-k)(n-1) \alpha_{n-1}^s + n(k-2) \alpha_n^s + \\
& \frac{k(n-1)}{2} \left(\frac{\alpha_n^s \alpha_{n-1}^b}{\alpha_n^b} + \alpha_{n-1}^s \right) = -\frac{3(h^s + l^s)(1-k)n h^b \alpha_n^b}{2((h^s)^2 + h^s l^s + (l^s)^2)} \\
\Rightarrow & \frac{3(h^s + l^s)(1-k)n h^b \alpha_n^b}{2((h^s)^2 + h^s l^s + (l^s)^2)} + (n-1) \left(k \frac{\alpha_n^s \alpha_{n-1}^b}{\alpha_n^b} + (1-k) \alpha_{n-1}^s \right) + n(k-2) \alpha_n^s + 1 = 0
\end{aligned}$$

Thus, for $j = n$, equating $\frac{\partial U^b}{\partial \alpha_n^b} = 0$ and $\frac{\partial U^s}{\partial \alpha_n^s} = 0$ would lead to following equations,

$$\frac{3(h^b + l^b)knl^s \alpha_n^s}{2((h^b)^2 + h^b l^b + (l^b)^2)} + (n-1) \left((1-k) \frac{\alpha_n^b \alpha_{n-1}^s}{\alpha_n^s} + k \alpha_{n-1}^b \right) - n(k+1) \alpha_n^b + 1 = 0 \quad (5.33)$$

$$\frac{3(h^s + l^s)(1-k)n h^b \alpha_n^b}{2((h^s)^2 + h^s l^s + (l^s)^2)} + (n-1) \left(k \frac{\alpha_n^s \alpha_{n-1}^b}{\alpha_n^b} + (1-k) \alpha_{n-1}^s \right) + n(k-2) \alpha_n^s + 1 = 0 \quad (5.34)$$

Finally, for any $j = i$ (where, $i \neq 1, n$), equating $\frac{\partial U^b}{\partial \alpha_i^b} = 0$ and $\frac{\partial U^s}{\partial \alpha_i^s} = 0$ would lead to following equations,

$$\begin{aligned}
\frac{\partial U^b}{\partial \alpha_i^b} = 0 &\Rightarrow \frac{1}{\alpha_i^s} - (i-1) \left(\frac{\alpha_{i-1}^b}{\alpha_{i-1}^s} - \frac{\alpha_i^b}{\alpha_i^s} \right) \left(\frac{(1-k)}{2} \frac{\alpha_{i-1}^s}{\alpha_i^s} \right) - (i-1) \left(-\frac{1}{\alpha_i^s} \right) \left(k \alpha_{i-1}^b + \frac{(1-k)\alpha_{i-1}^s}{2} \right. \\
&\quad \left. \left(\frac{\alpha_{i-1}^b}{\alpha_{i-1}^s} + \frac{\alpha_i^b}{\alpha_i^s} \right) \right) - i \left(\frac{\alpha_i^b}{\alpha_i^s} - \frac{\alpha_{i+1}^b}{\alpha_{i+1}^s} \right) \left(k + \frac{(1-k)}{2} \right) - \frac{i}{\alpha_i^s} \left(k \alpha_i^b + \frac{(1-k)\alpha_i^s}{2} \left(\frac{\alpha_i^b}{\alpha_i^s} + \frac{\alpha_{i+1}^b}{\alpha_{i+1}^s} \right) \right) = 0 \\
&\Rightarrow \frac{1}{\alpha_i^s} - \frac{(i-1)(1-k)}{2} \frac{\alpha_{i-1}^b}{\alpha_i^s} + \frac{(i-1)(1-k)}{2} \frac{\alpha_i^b \alpha_{i-1}^s}{(\alpha_i^s)^2} + (i-1)k \frac{\alpha_{i-1}^b}{\alpha_i^s} + \\
&\quad \frac{(i-1)(1-k)}{2} \frac{\alpha_{i-1}^b}{\alpha_i^s} + \frac{(i-1)(1-k)}{2} \frac{\alpha_i^b \alpha_{i-1}^s}{(\alpha_i^s)^2} - \frac{i(1+k)}{2} \frac{\alpha_i^b}{\alpha_i^s} + \\
&\quad \frac{i(1+k)}{2} \frac{\alpha_{i+1}^b}{\alpha_{i+1}^s} - ik \frac{\alpha_i^b}{\alpha_i^s} - \frac{i(1-k)}{2} \frac{\alpha_i^b}{\alpha_i^s} - \frac{i(1-k)}{2} \frac{\alpha_{i+1}^b}{\alpha_{i+1}^s} = 0 \\
&\Rightarrow \frac{1}{\alpha_i^s} + (i-1)k \frac{\alpha_{i-1}^b}{\alpha_i^s} + (i-1)(1-k) \frac{\alpha_i^b \alpha_{i-1}^s}{(\alpha_i^s)^2} - \frac{\alpha_i^b}{\alpha_i^s} \left(\frac{i(1+k)}{2} + ik + \frac{i(1-k)}{2} \right) + \\
&\quad \frac{\alpha_{i+1}^b}{\alpha_{i+1}^s} \left(\frac{i(1+k)}{2} - \frac{i(1-k)}{2} \right) = 0 \\
&\Rightarrow 1 + (i-1)(1-k) \frac{\alpha_{i-1}^s \alpha_i^b}{\alpha_i^s} + k(i-1) \alpha_{i-1}^b - i(1+k) \alpha_i^b + ik \frac{\alpha_i^s \alpha_{i+1}^b}{\alpha_{i+1}^s} = 0
\end{aligned}$$

$$\begin{aligned}
\frac{\partial U^s}{\partial \alpha_i^s} = 0 &\Rightarrow \frac{1}{\alpha_i^b} + (i-1) \left(\frac{\alpha_i^s}{\alpha_i^b} - \frac{\alpha_{i-1}^s}{\alpha_{i-1}^b} \right) \left(\frac{k}{2} \frac{\alpha_{i-1}^b}{\alpha_i^b} \right) + \frac{(i-1)}{\alpha_i^b} \left((1-k) \alpha_{i-1}^s + \frac{k \alpha_{i-1}^b}{2} \left(\frac{\alpha_i^s}{\alpha_i^b} + \frac{\alpha_{i-1}^s}{\alpha_{i-1}^b} \right) \right) + \\
&\quad i \left(\frac{\alpha_{i+1}^s}{\alpha_{i+1}^b} - \frac{\alpha_i^s}{\alpha_i^b} \right) \left(1 - k + \frac{k}{2} \right) + \frac{(-i)}{\alpha_i^b} \left((1-k) \alpha_i^s + \frac{k \alpha_i^b}{2} \left(\frac{\alpha_{i+1}^s}{\alpha_{i+1}^b} + \frac{\alpha_i^s}{\alpha_i^b} \right) \right) = 0 \\
&\Rightarrow \frac{1}{\alpha_i^b} - \frac{(i-1)k}{2} \frac{\alpha_{i-1}^s}{\alpha_i^b} + \frac{(i-1)k}{2} \frac{\alpha_i^s \alpha_{i-1}^b}{(\alpha_i^b)^2} + (i-1)(1-k) \frac{\alpha_{i-1}^s}{\alpha_i^b} + \frac{(i-1)k}{2} \frac{\alpha_{i-1}^s}{\alpha_i^b} + \\
&\quad \frac{(i-1)k}{2} \frac{\alpha_i^s \alpha_{i-1}^b}{(\alpha_i^b)^2} + \frac{i(2-k)}{2} \frac{\alpha_{i+1}^s}{\alpha_{i+1}^b} - \frac{i(2-k)}{2} \frac{\alpha_i^s}{\alpha_i^b} - i(1-k) \frac{\alpha_i^s}{\alpha_i^b} - \frac{ik}{2} \frac{\alpha_{i+1}^s}{\alpha_{i+1}^b} - \frac{ik}{2} \frac{\alpha_i^s}{\alpha_i^b} = 0 \\
&\Rightarrow \frac{1}{\alpha_i^b} + (i-1)(1-k) \frac{\alpha_{i-1}^s}{\alpha_i^b} + (i-1)k \frac{\alpha_i^s \alpha_{i-1}^b}{(\alpha_i^b)^2} - \frac{\alpha_i^s}{\alpha_i^b} \left(\frac{i(2-k)}{2} + i(1-k) + \frac{ik}{2} \right) + \\
&\quad \frac{\alpha_{i+1}^s}{\alpha_{i+1}^b} \left(\frac{i(2-k)}{2} - \frac{ik}{2} \right) = 0 \\
&\Rightarrow 1 + (i-1)k \frac{\alpha_{i-1}^s \alpha_i^b}{\alpha_i^b} + (i-1)(1-k) \alpha_{i-1}^s - i(2-k) \alpha_i^s + i(1-k) \frac{\alpha_i^b \alpha_{i+1}^s}{\alpha_{i+1}^b} = 0
\end{aligned}$$

Thus, for any $j = i$ (where, $i \neq 1, n$), equating $\frac{\partial U^b}{\partial \alpha_i^b} = 0$ and $\frac{\partial U^s}{\partial \alpha_i^s} = 0$ would lead to following equations,

$$1 + (i - 1)(1 - k) \frac{\alpha_{i-1}^s \alpha_i^b}{\alpha_i^s} + k(i - 1)\alpha_{i-1}^b - i(1 + k)\alpha_i^b + ik \frac{\alpha_i^s \alpha_{i+1}^b}{\alpha_{i+1}^s} = 0 \quad (5.35)$$

$$1 + (i - 1)k \frac{\alpha_{i-1}^b \alpha_i^s}{\alpha_i^b} + (i - 1)(1 - k)\alpha_{i-1}^s - i(2 - k)\alpha_i^s + i(1 - k) \frac{\alpha_i^b \alpha_{i+1}^s}{\alpha_{i+1}^b} = 0 \quad (5.36)$$

The above analysis is for generic n units k -double auctions consisting of a single buyer and a single seller in a single shot setting. Here, a player places a different bid for each of the units, and the corresponding scale factors can be found by solving the system of non-linear equations generated by equations 5.31, 5.33 and 5.35 for the buyer, and equations 5.32, 5.34 and 5.36 for the seller. The solution would result in a vector of scale factors for the buyer $\alpha^B = \{\alpha_1^b, \alpha_2^b, \dots, \alpha_n^b\}$, and a vector of scale factors for the seller $\alpha^S = \{\alpha_1^s, \alpha_2^s, \dots, \alpha_n^s\}$. However, for n unit of auctions, we need to find $2n$ number of unknowns (scale factors for the buyer and seller) by solving a system consisting of ***2n number of non-linear equations***, which is an ***NP-hard problem***.

5.4 Extending Theory to Practice: Designing Learning-based Bidding Strategy

We introduce an empirical approach to determine equilibrium strategies for PDAs. We describe our proposed bidding strategy, *DDPGBBS*, which empirically learns the equilibrium. We provide a comprehensive explanation of the strategy, including the training and validation setup for this novel approach. Additionally, we present experimental results demonstrating that *DDPGBBS* converges to the theoretical equilibrium. Finally, we extend our *DDPGBBS* implementation to make it applicable to generic PDAs with any number of buyers and sellers in incomplete information settings.

5.4.1 DDPGBBS: A Bidding Strategy for Multi-unit Auctions

In this section, we present the details of *DDPGBBS*. We focus on the scenario outlined in Section 5.3.2 to train the proposed strategy. The objective is to learn the buyer's Nash Equilibrium (NE) bidding strategy, given that the seller is playing its NE strategy. Specifically, we aim to determine the buyer's scale factors for each of the four cases discussed in Section 5.3.2. To achieve this, we propose the following MDP framework,

- ***State Space:*** It is a tuple consisting of *quantity to buy* (q) and *buyer's true type or valuation* (θ^b), denoted by $s = \{q, \theta^b\}$, where $q \in Q = \{0, 1, 2\}$, $\theta^b \in [0, 1]$, meaning the buyer's valuation for the item is between 0 to 1.
- ***Action Space:*** The actions are the buyer's scale-factors $a_1^b, a_2^b \in [0, 1]$, denoted by $a = \{\alpha_1^b, \alpha_2^b\}$.
- ***Reward:*** $r = 0$ if no market-clearing happens, otherwise $r = -\lambda * cq$ (where λ and cq are clearing price and buyer's clearing quantity).
- ***Transitions:*** A state s transition to the next state s' where *quantity to buy* is the remaining quantity ($q' = q - cq$) after auction-clearing in state s , ($q' \in Q = \{0, 1, 2\}$), while θ^b remains the same.
- ***Terminal State:*** As we consider a single-shot auction, MDP terminates after a single auction round. A buyer will receive a terminal reward $r^T = -q' * \theta^b$.

In our MDP model, which employs negative rewards, the optimal strategy is one that maximizes the expected reward. *DDPGBBS* learns by utilizing the clearing price information obtained from the auction clearing mechanism, which depends on the action taken by the agent, as shown in Figure 5.2. The reward function is designed such that if the buyer secures the items at a clearing price lower than their true valuation, they benefit; otherwise, they purchase the item at their true valuation, resulting in a higher clearing

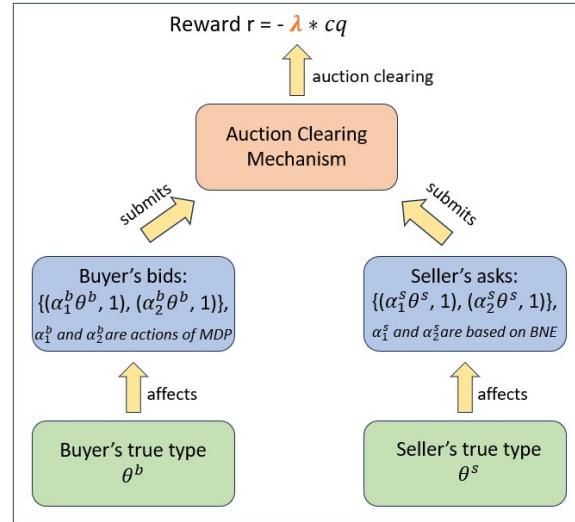


Figure 5.2: Relationship between Agent's Actions and Reward

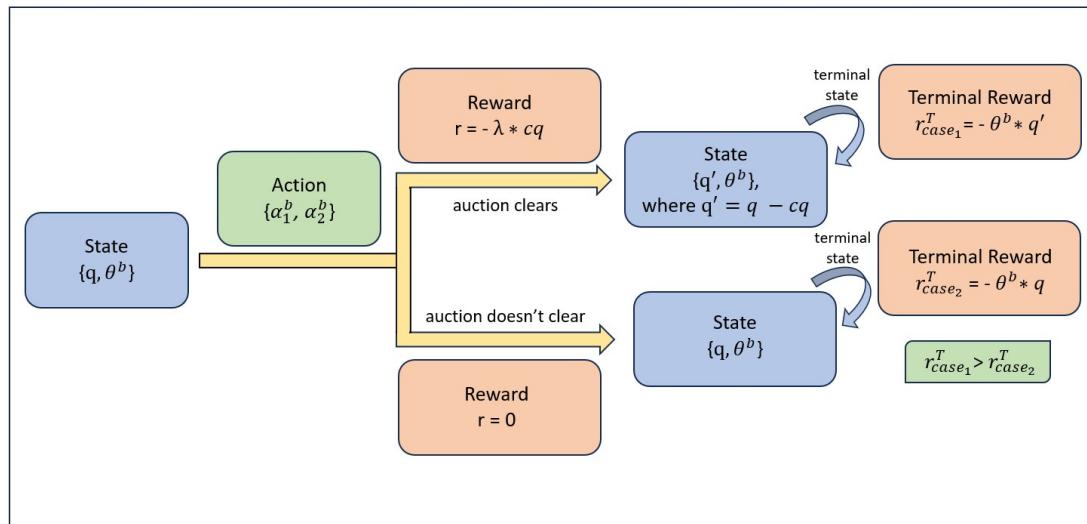


Figure 5.3: Transitions in of Two-stage MDP of DDPGBBS

price. This is demonstrated in Figure 5.3. *DDPGBBS*'s task is to determine the optimal scale factors that generate bids by analyzing the variation in rewards for the selected actions (scale factors). Lower bids increase profit (or reduce the cost to buy the item) but contain the risk of the item not getting cleared, while higher bids increase the likelihood of clearing but reduce profit (increase the cost to buy the item). *DDPGBBS* learns which scale factors optimize the buyer's expected utility.

The *DDPGBBS* architecture is as follows: the actor-network comprises two hidden layers with 40 and 30 units, respectively. The critic-network also consists of two hidden layers with 40 and 30 units, respectively, with actions not included until the second hidden layer. Both networks employ ReLU activation functions for all hidden layers. The actor-network uses a sigmoid activation function in the output layer, while the critic-network utilizes a linear activation function. The actor-network has two units in both the input and output layers. The neural network parameters are learned using the Adam optimizer, with learning rates of 10^{-4} for the actor and 10^{-3} for the critic. A $\gamma = 0.99$ discount factor is applied, and a soft target update rate of $\tau = 0.001$ is used. The choice of the hyperparameters is inspired by the DDPG paper [56], which has been validated in various environments. However, as our problem is relatively less complex (in terms of state space and action space dimensions), the number of units in the hidden layers is less than the original network and selected based on experiments.

5.4.2 Training and Validation Set-up of DDPGBBS

To train *DDPGBBS*, we designed a multi-unit k -double auction simulation incorporating the proposed scale-based bidding strategies for both the buyer and the seller. Our primary goal is to train *DDPGBBS* to learn the buyer's theoretical equilibrium; however, the same setup can be adapted to *learn the seller's theoretical equilibrium*. As detailed in Section 5.3.2, four possible scenarios are based on the buyer's and seller's scale factors,

with **Case-1** and **Case-3** leading to the same equilibrium and thus treated as a single scenario.

We train separate *DDPGBBS* models for the buyer’s equilibrium in each of the three distinct cases: **Case-1**, **Case-2**, and **Case-4**. In the simulation, the seller follows its scale-based equilibrium bidding strategy, with scale factors fixed according to theoretical results for each case, and its type is drawn from a uniform distribution $U \sim [0, 1]$. The buyer selects scale factors uniformly at random from the interval $[0, 1]$ and draws its type from $U \sim [0, 1]$, preparing bids using the scale-based bidding strategy.

The simulated auction clearing mechanism determines the clearing price and quantity. After each auction clearing, the buyer receives a reward and the next state, and the episode terminates. The buyer then prepares an experience quadruple (s_t, a_t, r_t, s_{t+1}) for each auction episode and stores it in a replay buffer. Random sampling from the replay buffer updates the policy and critic networks. We stop training after a specified number of iterations. In our experiments, we set the number of experiences to $1e5$ and the number of iterations to $1e4$.

5.4.2.1 Validating the Performance of DDPGBBS

Here we discuss the results of the validation experiments. As discussed above, there are three possible scenarios, and we trained a separate *DDPGBBS* model for each of the three cases. Table 5.4 compares buyer’s equilibrium scale factors learned by *DDPGBBS* against the theoretical scale factors for the case when $\alpha_{B1} = \alpha_{B2} = \alpha_B$ and $\alpha_{S1} = \alpha_{S2} = \alpha_S$. Similarly, Table 5.5 compares buyer’s equilibrium scale factors learned by *DDPGBBS* against the theoretical scale factors for the case when $\alpha_{B1} \neq \alpha_{B2}$ and $\alpha_{S1} = \alpha_{S2} = \alpha_S$. Finally, Table 5.6 compares buyer’s equilibrium scale factors learned by *DDPGBBS* against the theoretical scale factors for the general case when $\alpha_{B1} \neq \alpha_{B2}$ and $\alpha_{S1} \neq \alpha_{S2}$. Note that the buyer following *DDPGBBS* does not get to see the seller’s asks (neither during training nor during validation) and only learns the equilibrium strategy based on the feedback it

Table 5.4: Comparison b/w Experimental and theoretical scale factors of buyer **Case-1**

		α_{B1}
Theoretical Equilibrium		0.666667
DDPG Empirical Equilibrium	mean	0.577733
	std	0.023132

Table 5.5: Comparison b/w Experimental and theoretical scale factors of buyer **Case-2**

		α_{B1}	α_{B2}
Theoretical Equilibrium		0.857143	0.571428
DDPG Empirical Equilibrium	mean	0.928814	0.477797
	std	0.022228	0.046001

receives from the auction clearing mechanism for its own actions. These results demonstrate that *DDPGBBS* is able to detect the choice of seller's scale factors and effectively learn the NE strategy for each case.

From Table 5.4, we can see that the empirical result we obtained using *DDPGBBS* is within 12.2% of theoretical α_{B1} . From Table 5.5 (5.6), learned α_{B1} and α_{B2} are within 8.28%(3.13%) and 16.31%(47.6%) of the theoretical values, respectively. As we can see, except for α_{B2} in Table 5.6, *DDPGBBS* results are reasonably close to the theoretical results. As a comparison, the results presented by Susobhan et al. (in Table 1) [26] for single-unit k -double auction showed 20.19% and 24.72% difference between empirical results and theoretical values for buyer and seller, respectively. Additionally, *DDPGBBS* showed low variances for all the scale factors except α_{B2} in Table 5.6, reinforcing our *DDPGBBS*'s stability.

Table 5.6: Comparison b/w Experimental and theoretical scale factors of buyer **Case-3**

		α_{B1}	α_{B2}
Theoretical Equilibrium		0.882782	0.588521
DDPG Empirical Equilibrium	mean	0.855816	0.302352
	std	0.050594	0.221870

After validating the convergence of *DDPGBBS* to the theoretical equilibrium through experiments, we extend it for use in PDAs, which involve multiple auction rounds for a delivery slot, requiring modifications to *DDPGBBS*. Thus, to address the specific requirement for smart grid PDAs, we introduce Extended-*DDPGBBS*, along with experiments demonstrating its effectiveness.

5.4.3 Extending DDPGBBS to Operate in Sequential Multi-unit Double Auction of Smart Grids

As PDAs accommodate multiple auction instances for a delivery slot, we need to update *DDPGBBS* appropriately. To address this requirement, we introduce E-DDPGBBS, *tailored for a day-ahead PDA* of smart grids, as shown in Figure 5.4. The proposed strategy is an empirical extension of the analysis presented previously for scale-based strategies. It overcomes the limitations of the analysis by allowing any number of players to be included in PDAs. Moreover, it does not require the assumption of the players' known valuations. Below, we outline the revised MDP framework,

- **State space:** It is a tuple consisting of *proximity* (p), *quantity to buy at proximity p* (q^p) and *buyer's true type or valuation* (θ^b), denoted by $s^p = \{p, q^p, \theta^b\}$, where $p \in P = \{0, 1, 2, \dots, 24\}$, $q^p \in \mathbb{R}$, $\theta^b \in \mathbb{R}$. We consider the buyer's true type as *average unit balancing price for buying* from the balancing market.

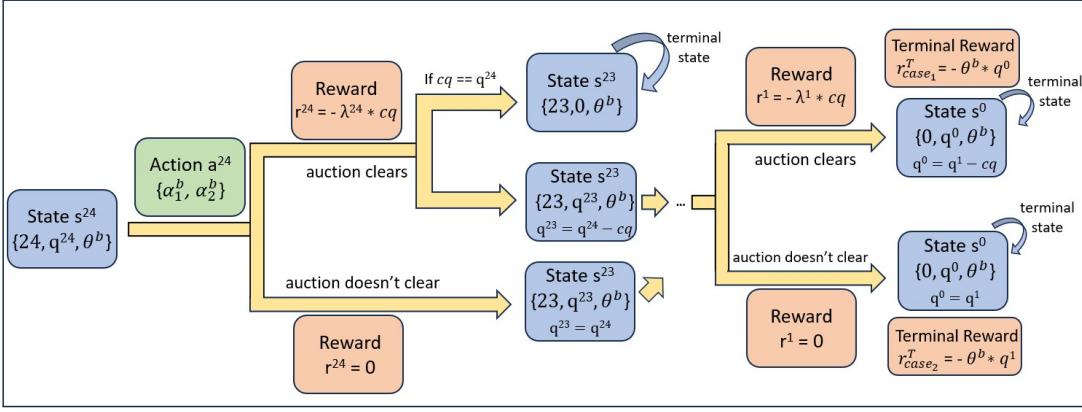


Figure 5.4: Transitions in of Multi-stage MDP of E-DDPGBBS

- **Action Space:** The actions at any proximity p are the buyer's scale-factors $\alpha^{b_1,p}, \alpha^{b_2,p} \in [0, 1]$, denoted by $a^p = \{\alpha^{b_1,p}, \alpha^{b_2,p}\}$.
- **Reward:** $r^p = 0$ if no market-clearing happens, otherwise $r^p = -\lambda^p * cq^p$ (where λ^p and cq^p are clearing price and buyer's clearing quantity at proximity p).
- **Transitions:** A state s^p transition to the next state s^{p-1} with *proximity* $p-1$ where *quantity to buy* is the remaining quantity ($q^{p-1} = q^p - cq^p$) after auction-clearing in state s^p , ($q^{p-1} \in \mathbb{R}$), while θ^b remains the same.
- **Terminal State:** MDP terminates when $p = 0$ or $q^p = 0$. A buyer will receive a terminal reward $r^T = -q^p * \theta^b$.

During gameplay, E-DDPGBBS outputs two scale factors that get multiplied by the buyer's true valuation to generate two bids for the auction. The required bidding quantity is equally divided between these two bids. Here, too, given negative rewards, the optimal strategy aims to maximize the expected reward. It is easy to see that this is an extension of DDPGGBS.

We train E-DDPGBBS offline by gathering experiences in a replay buffer using the PowerTAC PDA simulator. We conduct two sets of experiments, each consisting of 20

games. In the first set, E-DDPGBBS competes against a ZI broker in two-player games. In the second set, E-DDPGBBS engages with three other ZI brokers in four-player games. In both sets, hourly demand is evenly distributed among all participating brokers, ensuring each broker equally engages in the wholesale market PDA. After each auction instance in the game, E-DDPGBBS updates the replay buffer. We chose to train against ZI brokers because they exhibit a broad spectrum of bidding behaviours, which exposes E-DDPGBBS to diverse states within the state space S , thereby enhancing its learning capabilities. Upon completing both sets of experiments, we consolidate the experiences from the combined replay buffer and update E-DDPGBBS using the standard DDPG update procedure, consistent with our validation experiments.

5.4.3.1 Experiments and Results

Here, we analyze the efficacy of Extended *DDPGBBS* in PDA by leveraging the PowerTAC PDA simulator. We first explain the experimental setup, followed by the results of the offline experiments. To compare the performance of Extended *DDPGBBS*, we isolate the PowerTAC wholesale market while keeping the default market participants. We benchmark the performance of Extended *DDPGBBS* against the state-of-the-art and baseline strategies mentioned in Section 5.2.

We perform two batches of experiments; the first batch of experiments is divided into four sets. In each of these four sets, we play ten two-player games where Extended *DDPGBBS* is one of the two brokers in each set, while the second broker is selected from the list of four benchmarking brokers. In other words, Extended *DDPGBBS* competes against each benchmarking broker in a set of 10 two-player games. Similarly, in the second batch, we play ten five-player games with all the available brokers in the game. The number of games is selected based on empirical observations; a total of 14400 auctions are conducted in 10 games, which is sufficient for comparison. Table 5.7 summarises the results of the first batch of experiments; it shows the normalized average unit clearing price of each set

Table 5.7: Relative Unit Clearing Price Comparison

E-DDPGBBS	SPOT	VV	ZI	ZIP
1.0	1.2832	1.0992	1.3376	1.1920

with respect to the Extended ddpg’s clearing price. A value greater than 1 indicates that the competing broker in that set had a higher average clearing price than Extended *DDPGBBS* after playing ten games. Based on the results in Table 5.7, we observe that Extended *DDPGBBS* outperforms all the other bidding strategies consistently by at least 9.9% in two-player games while achieving 33.76% improvement against ZI.

Figure 5.5 shows the result for the second batch of experiments, where we compare the average unit clearing price of each broker across ten games in a five-player game configuration. The error bars on top of the bar plots show the standard deviation of average clearing prices in 10 games. Here, too, Extended *DDPGBBS* consistently outperforms all the other brokers by 21.42% against second-performing VV while achieving almost 46% improvement against other brokers. *DDPGBBS* obtained better results than the state-of-the-art strategies (VV and SPOT) used for comparison. *DDPGBBS* improves in the range [21.42%, 46%] 5.5 over the baseline strategies as well as over VV and SPOT. Susobhan et al.’s VV (in Figure 1 [26]) improved in the range [-11.46%, 22.01%], while Chowdhury et al. (in Figure 3 in [81]) improved in the range [31.28%, 32.7%] over previous strategies. It is also to be noted that, unlike some of the baseline brokers, Extended *DDPGBBS* does not incorporate any additional heuristics in its bidding strategy and still outperforms some of the best brokers of PowerTAC.

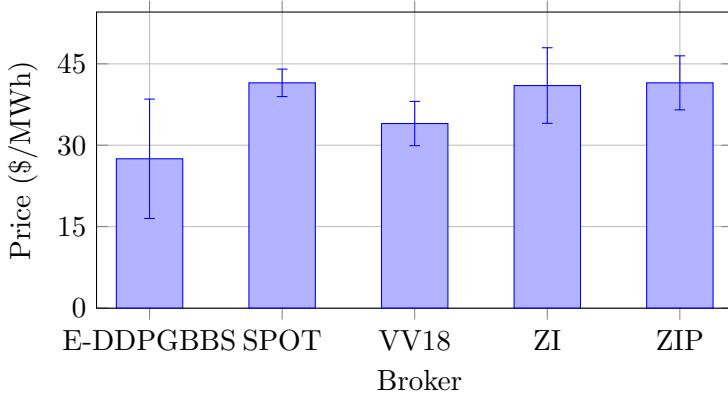


Figure 5.5: Average Unit Clearing Price Comparison for five-player game configuration

5.5 Summary

In this chapter, we first characterized the Bayesian Nash equilibrium for single-buyer, single-seller, two-unit single-shot k -double auctions, where the buyer and the seller follow scale-based bidding strategies. We analyzed all possible cases resulting from how the buyer and the seller select their scale factors and presented a set of non-linear equations for a general case. As the equilibrium analysis becomes intractable with the increase in the number of participating players and items, we presented a deep deterministic policy gradient bidding strategy *DDPGBBS* that is easily extendable to a real-world PDA. We experimentally validated that *DDPGBBS* achieves BNE approximately for each case. Furthermore, we proposed an extension of *DDPGBBS* that can be utilized in the real-world PDA. Finally, to examine the Extended *DDPGBBS* efficacy, we benchmarked it against several baselines and the state-of-the-art bidding strategy of PowerTAC wholesale PDA. We showed that it consistently outperforms some of the best PowerTAC wholesale bidding strategies.

Chapter 6

Designing Bidding Strategies via Learning Supply Curve

Having seen a scale-based bidding strategy backed up by equilibrium theory to place bids in the smart grid wholesale market in the last chapter, it is imperative that any bidding strategy needs to be real-time to be effective in the wholesale market auctions. In this chapter, we propose another real-time bidding strategy that utilizes the information available to the broker during the games. Precisely, the proposed bidding strategy aims to model the supply curve or the cost curve of the prominent GenCo in the wholesale market and use the **supply curve model** to place bids in the auction depending on the number of opportunities left in the auction; we call this strategy, **SupplyCurveBBS**. The SUPPLYCURVEBBS aims to reduce the cost of procurement by following the supply curve of GenCo to identify its lowest ask for each auction, which is then used to generate suitable bids. We further demonstrate the efficacy of SUPPLYCURVEBBS in PowerTAC2021 finals and through several controlled experiments¹.

¹This work has been published as a part of IJCAI 2022 paper.

6.1 Introduction

The importance of double auctions and the impact of AI-based bidding strategies for wholesale trading is already discussed in the previous chapter. We also discussed the challenges one might face while designing a bidding strategy for a PDA, as the broker needs to plan for the current as well as future auction instances. Furthermore, as discussed in the previous chapter as well as in Section 2.2.3, there are multiple types of auctions, and each type of auction has its own characteristics and information available that can be used to design a bidding strategy. Thus generated strategies need to be *real-time* as well to be useful in the auctions. Considering all the points mentioned above, we propose a strategy that uses the uncleared ask of GenCos to model the supply curve or the cost function of GenCo. In PowerTAC, each broker receives information about all the unclear asks and unclear bids of the previous auction instance from the game server, which we utilize in our proposed strategy.

In this chapter, we describe the design of our *supply curve-based bidding strategy*, **SupplyCurveBBS**. The same method can be extended to any real-world GenCo. The proposed bidding strategy aims to place bids by identifying the *lowest ask of the largest seller* by following its supply curve. This ask price is used as an upper bound to procure as much power from other sellers present in the wholesale market at lower prices. This strategy also instils risk-taking ability in the broker. In SUPPLYCURVEBBS, the broker places bids slightly lower than the identified GenCo's ask to purchase the electricity from other brokers at a price lower than the GenCo during the initial auction opportunities. However, as the auction approaches balancing, the broker becomes conservative and bids near GenCo's lowest ask to purchase the electricity from the wholesale market itself. Thus, the proposed strategy attempts to procure all the required electricity from the wholesale market, thus enabling effective electricity procurement and reducing imbalance costs.

The modelling of GenCo's supply curve from the available information does not involve complex computation and thus makes the bidding strategy real-time and easily deployable.

Furthermore, as the PowerTAC game environment may vary drastically from game to game, the modelling of the supply curve is done from scratch in each game, and thus, the strategy is *highly adaptive* from game to game. Because of the above characteristics, we preferred the SUPPLYCURVEBBS to deploy in our broker, VidyutVanika21 (VV21) in the PowerTAC2021 tournament. Finally, here too, we take PowerTAC’s wholesale market PDA as a testbed to evaluate our novel bidding strategy. We benchmark SUPPLYCURVEBBS against several baselines and state-of-the-art bidding strategies of the PowerTAC wholesale market PDA and demonstrate its efficacy in the annual PowerTAC2021 tournament.

6.2 Preliminaries and Mathematical Model

In order to explain the SUPPLYCURVEBBS, we present the full wholesale module of our broker VidyutVanika21. The wholesale module of VidyutVanika21 (VidyutVanika21-WM) aims to procure the electricity needed for its subscriber base at the lowest possible cost. To this end, first, VidyutVanika21-WM uses the *demand predictor (DP)* sub-module to estimate the expected demand (Q_T) of its subscriber base for a delivery time slot T . After that, the *bid generator (BG)* sub-module participates in periodic double auctions (PDA) by placing bids of the form (p, q) where q is the amount of electricity sought to be procured at no more than price p . Recall that in the PowerTAC setting, for a delivery time slot T , a broker can start procuring electricity from time slot $T - 24$ at hourly intervals. Hence, BG has 24 opportunities to buy the required electricity. This also implies that at any bidding time slot t , BG participates simultaneously in 24 auctions corresponding to 24 future (delivery) time slots denoted by $(t + 1, t + 2, \dots, t + 24)$. BG can submit multiple bids for a single auction. The *supply curve follower (SCF)* sub-module aids BG in determining suitable limit prices for bids that are to be placed in an auction. Any shortfall or excess in electricity procurement by BG after exhausting all 24 opportunities is adjusted in the balancing market, which is typically expensive for the broker. We describe SCF and BG sub-modules here and use the PowerTAC sample broker’s predictor as a DP sub-

module. Note that although the wholesale module is described from a buyer’s perspective, it is straightforward to design a similar behaviour for brokers who wish to sell electricity using production customers in their portfolio. However, as in this work, our broker does not buy excess electricity and thus does not place any sale orders.

6.3 Design of SupplyCurveBBS

In this section, we present the details of SUPPLYCURVEBBS along with all the other sub-modules of VidyutVanika21-WM. More precisely, we present three sub-modules: the supply curve follower (SCF), the bid generator (BG), and the demand predictor (DP). The demand predictor module is used to predict customers’ usage as well as the market’s net demand. The SCF sub-module is the core of the SUPPLYCURVEBBS, while other sub-modules provide assistance to make the strategy function.

6.3.1 The Supply Curve Follower (SCF) Module

The SCF helps determine suitable limit prices for generating bids. In the PowerTAC wholesale market, a GenCo acts as the main supplier, and the relationship between supply and demand (p, q) is *quadratic* in nature. The core idea involves constructing a supply-demand curve through uncleared asks and following the curve to *identify the minimum ask price of GenCo* for given outstanding electricity requirements of all buyers. Each auction can be marked using the tuple (t, T) where $t \in \{T - 24, \dots, T - 1\}$ are the bidding times and T is the delivery time. VV21-WS does not place any bids at the first opportunity (i.e., at $t = T - 24$) because there is no information available about the supply curve of the seller. When the uncleared asks are empty, the least ask price is the maximum of the market-clearing prices of past auctions where the difference between the delivery time and bidding time is $T - t$. Algorithm 11 outlines SCF.

Supply Curve Follower

Algorithm 11: Supply_Curve_Follower(*currentTime*, *netDemand*)

Input: Current time *currentTime* and net demand *netDemand*

Output: list of ask prices *askPrices[]*

```

1: for hour in [1 . . . 23] do
2:   futureTime = currentTime + hour
3:   Get UnclearedAsks of Auction(currentTime-1, futureTime)
4:   if UnclearedAsks is empty then
5:     Get ClearedPrices of Auction(t, t+hour)  $\forall t < \text{currentTime}$ 
6:     askPrices[futureTime] = max [ClearedPrices]
7:   else
8:     Sort UnclearedAsks ( $p_i, q_i$ ) such that  $p_i < p_{i+1}$ 
9:     brokerBalance = energyBought(broker, futureTime)
10:     $\hat{q} = \text{netDemand}[futureTime] - \text{brokerBalance}$ 
11:    askPrices[futureTime] = min  $p_r$  s.t  $\hat{q} \leq \sum_{i=0}^r q_i$ 
12:   end if
13: end for
14: return askPrices[]

```

6.3.2 The Bid Generator (BG) Module

The BG is responsible for *generating multiple bids* (M bids in total) using the outputs of DP and SCF. Based on empirical observation, BG recognizes the possibility of procuring electricity at a lower price than GenCo's price as other smaller sellers (like opponent brokers) sell their excess procurement in the wholesale market. However, predicting the exact time and asking prices of such sales is difficult. Hence, when the bidding time t is far from the delivery time T , BG generates bids with limit prices less than the esti-

mated least ask of the GenCo (using hyperparameters α_f, β_f), hoping to capitalize on asks from smaller sellers. Closer to the delivery slot, BG generates bids with limit prices that are slightly higher than the least ask of the GenCo (using hyperparameters α_c, β_c). This ensures procurement of the outstanding electricity from GenCo and avoids going to the balancing market where electricity costs are typically high. Algorithm 12 describes the working of BG.

Bid Generator

Algorithm 12: Bid_Generator(*currentTime*, *askPrices*)

Input: Current time *currentTime* and list of ask prices *askPrices*)

Output: list of bids *bids*

```

1: estUsage[] = DemandPredictor(currentTime)
2: for hour in [1,2...23] do
3:   futureTime = currentTime + hour
4:   if (currentTime is far from futureTime) then
5:     minPrice =  $\alpha_f * \text{askPrices}[\text{futureTime}]$ 
6:     maxPrice =  $\beta_f * \text{askPrices}[\text{futureTime}]$ 
7:   else
8:     minPrice =  $\alpha_c * \text{askPrices}[\text{futureTime}]$ 
9:     maxPrice =  $\beta_c * \text{askPrices}[\text{futureTime}]$ 
10:  end if
11:  Sample M prices in the range [minPrice,maxPrice]
12:  Distribute estUsage[futureTime] uniformly across M prices
13:  Create bids  $(q_i, p_i) \forall i \in [1, \dots, M]$  into bids[futureTime]
14: end for
15: return bids[][]

```

6.3.3 The Demand Predictor (DP) Module

Demand Predictor

Algorithm 13: Demand_Predictor(*currentTime*)

Input: Current time *currentTime*

Output: Demand forecast *estUsage*[]

```

1: weekHour = getHourOfWeek(currentTime)
2: for cust in broker.customerModels do
3:   cust.numSubs = getCurrentSubscriptions(cust)
4:   cust.avgUsage[weekHour] =  $\gamma * cust.avgUsage[weekHour] + (1 - \gamma) *$ 
      getCurrentUsage(cust)
5: end for
6: for hour in [1 ... 24] do
7:   futureTime = currentTime + hour
8:   weekHour = getHourOfWeek(futureTime)
9:   estUsage[futureTime] = 0.0
10:  for cust in broker.customerModels do
11:    estUsage[futureTime] += cust.avgUsage[weekHour] * cust.numSubs
12:  end for
13: end for
14: return estUsage[]
```

The Demand Predictor sub-module is used to *predict the expected usage* of VV21's subscriber base up to 24 hours into the future. To predict that, DP maintains a 168-dimensional usage array for each customer corresponding to every hour of the week. This array is initialized with the average usage of a customer from the boot period. Thereafter, as and when the broker obtains a new observation pertaining to customer consumption for a particular hour of the week, the corresponding entry of the usage array is updated using a simple moving average rule. The result is then multiplied by the current number of subscriptions of that customer, and the total usage for a delivery slot is then obtained

by summing the predicted usage across all customers in the retail portfolio. The details are present in Algorithm 13.

The prediction of the net market demand (NMD), which is the total sum of the usage patterns of all customers in the retail market, is also made using the moving average method. Algorithm 14 outlines the details of NMD prediction.

Net Demand Predictor

Algorithm 14: Net_Demand_Predictor(*currentTime*)

Input: Current time *currentTime*

Output: Net demand forecast *NetDemand*[]

```

1: weekHour = getHourOfWeek(currentTime)
2: prod.avgVal[weekHour] =  $\gamma$  * prod.avgVal[weekHour] + (1 -  $\gamma$ ) *
   getMarketProduction(currentTime)
3: cons.avgVal[weekHour] =  $\gamma$  * cons.avgVal[weekHour] + (1 -  $\gamma$ ) *
   getMarketConsumption(currentTime)
4: for hour in [1 . . . 24] do
5:   futureTime = currentTime + hour
6:   weekHour = getHourofWeek(futureTime)
7:   NetDemand[futureTime] = cons.avgVal[weekHour] - prod.avgVal[weekHour]
8: end for
9: return NetDemand[]

```

6.4 SupplyCurveBBS in PowerTAC: Experiments and Results

As the SUPPLYCURVEBBS is designed specifically to deploy in the PowerTAC tournament, there are no further modifications required to make it suitable for real-world smart grid systems. In this section, we perform validation experiments to verify the efficacy of the

strategy. We present the experimental set-up, followed by results and discussion. Finally, we showcase the effectiveness of SUPPLYCURVEBBS in the PowerTAC2021 tournaments.

6.4.1 Experimental Set-up

In this experiment setup, we analyze the efficacy of SUPPLYCURVEBBS in wholesale and balancing markets with the best wholesale strategies of brokers from previous PowerTAC tournaments. In this analysis, we play a mini-tournament of 30, 42, and 70 games in 7, 5, and 3-player configurations, respectively. The number of games and choice of player configurations come from the PowerTAC 2021 qualifiers, where a similar setup was adopted for the competition. The games are also played across different real-world locations with modified brokers of VV21, AgentUDE (A), TUC-TAC (TT), Crocodile (C), VidyutVanika18 (VV18), TacTex (TX) and Spot (ST). The wholesale strategy of each competing broker is retained as in the original bots, but the tariff strategy is replaced with constant fixed price tariffs for all kinds of customers to maintain a common reference point. Below, we present the results of these experiments and provide a detailed discussion of the results.

6.4.2 Results and Discussion

The performance of the brokers in the wholesale market is analyzed by computing the *mean wholesale price (MWP)* for buying unit MW of electricity in different game configurations. The results are shown in Tables 6.1 and 6.2, which indicate that SUPPLYCURVEBBS is still the dominant strategy in most different settings against popular wholesale strategies. More specifically, Table 6.1 shows the mean wholesale price for buying a unit of electricity for all the brokers in three different player configurations, namely, 3-Player, 5-Player, and 7-Player. VV21 is able to purchase electricity at the lowest prices among all the brokers in all three player configurations. Furthermore, Table 6.2 exhibits the mean wholesale price for buying a unit of electricity for all the brokers in three different weather conditions of Denver, Minneapolis, and Phoenix regions. In this experiment too,

Table 6.1: Mean Wholesale Prices of Brokers for each Player Configuration

Broker	3-Player	5-Player	7-Player
VV21	-58.52	-48.92	-46.03
C	-121.81	-86.03	-81.08
ST	-71.16	-58.50	-50.12
TT	-51.22	-53.78	-59.27
TX	-89.53	-93.62	-75.02
A	-99.51	-67.08	-79.35
VV18	-67.50	-120.49	-133.46

VV21 emerged as the broker with the lowest purchase cost in two regions except Phoenix. Notably, during the Phoenix games, the wholesale purchase cost of VV21 is staggeringly different from that of other brokers. Only TT has a better wholesale purchase cost than VV21. Note that the *Phoenix region* is introducing unexpectedly *high customer demands*. Due to this, the market ends up in deficit as GenCos are not able to supply the required electricity to fulfil extraordinary demands, and thus, wholesale prices shoot up. SUPPLYCURVEBBS is able to procure electricity at cheaper rates in such adverse scenarios as well, which showcases the robustness of the bidding strategy.

6.4.3 SupplyCurveBBS in PowerTAC2021 Tournament

As mentioned earlier, we deployed the SUPPLYCURVEBBS as our wholesale market strategy in our broker VidyutVanika21 for the PowerTAC2021 tournament. In the previous section, we showcased the performance of SUPPLYCURVEBBS in the validation experiments that motivated us to deploy it in the tournament. Further analysis of the PowerTAC2021

Table 6.2: Mean Wholesale Prices of Brokers for each Weather Condition

Broker	Denver	Minneapolis	Phoenix
VV21	-42.43	-46.12	-67.73
C	-71.87	-66.69	-133.23
ST	-44.85	-47.37	-98.78
TT	-54.97	-54.78	-52.29
TX	-64.79	-61.34	-114.63
U	-67.12	-73.60	-117.33
VV18	-85.81	-81.74	-118.03

tournament data establishes that SUPPLYCURVEBBS has the lowest mean market price in electricity procurement from the wholesale and balancing market with respect to different player configurations and different weather locations, as shown in Table 6.3 and 6.4, respectively. On closer inspection of Table 6.3, we observe that VV21’s mean wholesale price for procuring one unit of electricity is almost half of the second-best broker in all three player configurations; thus, VV21 was able to buy electricity at half the price than the other brokers in the tournament. Similarly, as shown in Table 6.4, VV21 had a significantly lower mean wholesale price to procure a unit of electricity than the second-best broker in all three weather regions. Notably, in the Phoenix region, VV21 was able to procure electricity at astonishingly lower prices than any other broker in the market. Our analysis shows that the ability of SUPPLYCURVEBBS to handle adverse market conditions in Phoenix games contributed significantly to curtailing losses of VV21 in the tournament and helped it to be the champion. The overall mean market price of SUPPLYCURVEBBS in the tournament is $-54.87\$/MW$, which is relatively 79% cheaper than the price of the next cheapest broker.

Table 6.3: Mean Wholesale Prices each Player Configuration in PowerTAC2021

Broker	3-Player	5-Player	7-Player
CP	-148.5	-252.2	-134.1
C	-97.42	-97.47	-115.9
M	-127.8	-143.6	-182.4
TT	-126.2	-126.9	-110.2
VV21	-59.76	-46.63	-55.11

Table 6.4: Mean Wholesale Prices each Weather Condition in PowerTAC2021

Broker	Denver	Minneapolis	Phoenix
CP	-103.1	-113.1	-318.4
C	-85.44	-99.32	-123.4
M	-69.42	-89.03	-219.4
TT	-82.32	-92.94	-214.2
VV21	-55.72	-60.78	-39.17

Finally, Figure 6.1 shows the overall average unit clearing price comparison of all the brokers in PowerTAC2021 across all player and weather configurations. As depicted, VV21 achieved the lowest average wholesale cost, which is nearly half that of the second-best broker, highlighting the effectiveness of the proposed strategy.

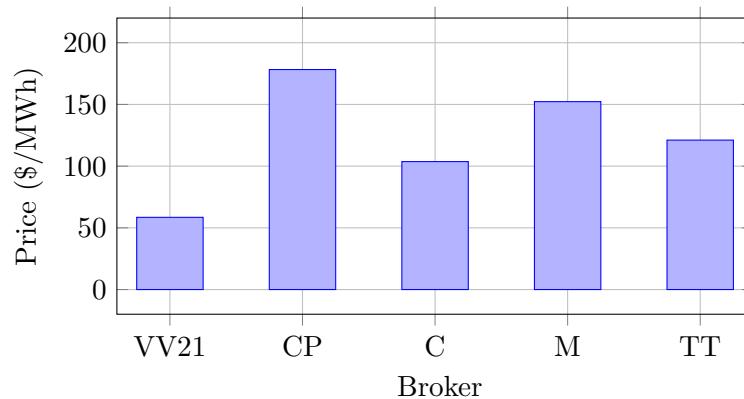


Figure 6.1: Overall Average Unit Clearing Price Comparison for in PowerTAC2021

6.5 Summary

In this chapter, we discussed the design of a wholesale market bidding strategy SUPPLYCURVEBBS, which uses the information made available by the game server to acquire the knowledge of the seller's bidding pattern in the wholesale market and use it to procure power at lower prices and avoid making purchases in the balancing market. We presented the working of the wholesale module of VV21 where SUPPLYCURVEBBS acts as a core component. Lastly, we presented a detailed analysis of its performance with the help of validation experiments as well as in the PowerTAC2021 tournament.

Chapter 7

Designing Bidding Strategies via Markov Perfect Nash Equilibrium Analysis

In Chapter 5, we conducted an equilibrium analysis of scale-based bidding strategies in single-shot double auctions, focusing on single-buyer and single-seller settings by framing the strategic interaction as a two-player game. However, as the complexity increases—whether due to a larger number of participants or the transition to periodic double auctions (PDAs) with incomplete information—the tractability of this analysis diminishes. Acknowledging these limitations, this chapter introduces a new theoretical framework by modelling PDAs as a **Markov game**, enabling an equilibrium analysis applicable to settings with any number of participants. We derive **Markov perfect Nash equilibrium (MPNE)** policies for scenarios where multiple buyers compete for commodities, leveraging the composite supply curve of sellers. To address incomplete information, buyers approximate the composite supply curve using historical auction data, which underpins the

development of the **MPNE-BBS** algorithm for PDAs. This algorithm is specifically designed to optimize procurement strategies under uncertainty.

To validate the proposed theory, we apply it to PDAs in the smart grid wholesale market as a use case. We evaluate MPNE-BBS using the Power-TAC wholesale market PDA simulator, comparing its performance against existing benchmark strategies. The results demonstrate the algorithm's effectiveness in minimizing procurement costs, highlighting its advantages over baseline strategies¹.

7.1 Introduction

As discussed in the previous chapters, a periodic double auction (PDA) is a setup where buyers and sellers participate in a series of auctions to trade units of a commodity. PDAs are particularly prevalent in energy markets, where power-generating companies (GenCos) serve as prominent sellers and retail brokers act as buyers. The periodic nature of these auctions allows participants to trade energy incrementally until just a few hours before delivery. However, designing an effective *bidding strategy* that accounts for both current and future auctions presents a significant challenge. To address this, we model PDAs as a *finite horizon Markov game* and derive *equilibrium solutions* to enable efficient bidding strategies.

The development of efficient bidding strategies is largely influenced by the auction's clearing and payment rules. While equilibrium solutions for double auctions have been extensively studied under various payment mechanisms [129], including existence results for the average clearing price rule [25], as well as multi-buyer [26] and multi-item settings [27], these works are limited to single-shot double auctions. Such results are not directly applicable to PDAs, where the sequential nature of decision-making introduces additional complexities.

¹This work has been published at IJCAI 2024, and the extended work is under review at AIJ.

Recent studies have attempted to model PDAs using Markov games and reinforcement learning techniques, such as multi-agent Q-learning [130], multi-agent deep Q-networks [131], and deep deterministic policy gradients (DDPG)[132, 27]. However, many of these works focus on single-sided auctions² and rely heavily on numerical simulations rather than providing analytical solutions. A notable recent work[133] addressed PDAs by modelling them as complete information Markov games, proposing an equilibrium solution under the *average clearing price rule (ACPR)*. However, this approach is limited to idealized conditions, where supply from wholesale suppliers is always adequate, and it does not extend to realistic settings involving incomplete information.

In this work, we advance the state of the art by leveraging the Markov game framework to derive equilibrium solutions that form the basis for robust bidding strategies under dynamic and realistic market conditions. Specifically, we begin by modelling PDAs as a finite horizon Markov game and propose equilibrium strategies for settings with complete information. Unlike previous approaches, our framework generalizes to uniform pricing payment rules that satisfy specific desirable properties. To address incomplete information scenarios, we extend our approach by introducing a novel bidding strategy, MPNE-BBS, which utilizes historical auction data to approximate the composite supply curve. This enables buyers to make well-informed bidding decisions, bridging the gap between theory and practical market dynamics.

We evaluate the proposed MPNE-BBS strategy within the smart grid energy market using the PowerTAC simulator. The performance of MPNE-BBS is rigorously tested across various market conditions, comparing it against multiple baselines and state-of-the-art methods. The experimental results demonstrate the effectiveness of our strategy in optimizing procurement costs. Finally, we provide a comprehensive discussion of the strengths and limitations of the proposed bidding strategies, offering insights into their applicability in real-world market settings.

²Auctions where either buy bids or sell bids are placed, but not both.

7.2 Preliminaries

We have already defined the auction setting and the notations in Chapter 5; we repeat them here for the sake of completeness. We consider a market with B buyers competing to procure multiple units of a commodity from S sellers by participating in a PDA with H rounds. In any round $h \in [H]$ ³ of the PDA, a buyer $b \in [B]$ has $Q^{b,h}$ units of a commodity to purchase, with each $Q^{b,h}$ being unique for every buyer $b \in [B]$. The set of bids for buyer b in round h is denoted as $\mathcal{B}^{b,h}$, containing $|\mathcal{B}^{b,h}|$ elements. Each bid is represented as a pair of price and quantity $(p_i^{b,h}, q_i^{b,h})$, where $i \in [| \mathcal{B}^{b,h} |]$, with the price and quantity bounded by p_{\max} and $Q^{b,h}$, respectively. Consequently, the total outstanding demand in round h from all B buyers is given by $Q^{\mathcal{D},h} = \sum_{b \in [B]} Q^{b,h}$. Furthermore, we assume that the buyers cannot purchase more units of a commodity than their requirement.

Conversely, the sellers are represented by a consolidated supply curve with $|\mathcal{L}^h|$ asks, expressed as $\mathcal{L}^h = \{(p_i^h, q_i^h); i \in [| \mathcal{L}^h |]\}$, where $p_i^h \in [0, p_{\max}]$ and $q_i^h \in [0, q_{\max}]$ are the price and quantity components of the i^{th} ask (p_i^h, q_i^h) , with p_{\max} and q_{\max} serving as suitable upper bounds. The total supply provided by wholesale sellers in round h is denoted as $Q^{\mathcal{S},h} = \sum_{i \in [| \mathcal{L}^h |]} q_i^h$.

At each round h , the market regulator collects the elements of \mathcal{L}^h and \mathcal{B}^h and applies a **clearing rule** to determine the clearing price(s) and quantities. This work focuses on **uniform pricing** as the payment rule, where all cleared bids and asks have the same clearing price. The cleared price for round h is referred to as the market clearing price, denoted by λ^h , and the total quantity cleared for all the $|\mathcal{B}^h|$ bids in round h is denoted as Q^h . To elucidate the clearing process, we assume, without loss of generality, that the elements of the set \mathcal{L}^h (\mathcal{B}^h) are sorted in increasing (decreasing) order of their price components. If the price components are equal, then \mathcal{L}^h and \mathcal{B}^h are sorted in decreasing order of their quantity components. If the price and quantity components are identical, the ordering between them is arbitrary. We adopt uniform pricing as the payment rule,

³We denote $[K]$ as the set $1, \dots, K$ for any integer K .

ensuring that all winning buyers and sellers pay or receive the same price per unit of the commodity traded, dependent on the market clearing rule. One of the most prominent clearing rules for k -double auctions is defined in Section 4.2.2.1.

7.3 The Markov Game Structure

Consider a Markov game [134] $\mathcal{G}\langle N, S, A, C, P, H \rangle$ with a finite horizon, designed to model the PDA involving N players, where the players are B buyers. The state space S is defined as the set of sellers' asks \mathcal{L}^h and buyers' demand \mathcal{Q}^h , such that $S = \{\mathcal{L}^h, \mathcal{Q}^h\}$. Here, \mathcal{Q}^h denotes the demand $\mathcal{Q}^{b,h}$ of all buyers $b \in [B]$. Operating in a *complete information setting*, both the sellers' asks and buyers' demands are known to all buyers, making the state transitions deterministic.

The joint action space $A = \times_{b \in [B]} A^b$ is the Cartesian product of all buyers' action spaces A^b , where $A^b = \cup_{h \in [H]} \mathcal{B}^{b,h}$ represents the product space of all bids submitted by a buyer over $h \in [H]$. The cost function $C^{b,h} : S \times A^b \rightarrow \mathbb{R}$ for a player b is then defined as,

$$C^{b,h}(s^h, a^h) = \begin{cases} \sum_{i \in [|\mathcal{B}^{b,h}|]} \lambda^h \cdot c q_i^{b,h} & 1 \leq h \leq H \\ \Upsilon \times Q^{b,h} & h = H + 1, \end{cases}$$

where $a^h = (a^{b,h}, a^{-b,h}) \in A$ represents the joint action, $a^{-b,h}$ denotes the actions taken by players other than b , $c q_i^{b,h}$ represents the cleared quantities for player b 's i th bid, and λ^h is the clearing price in round h . The constant $\Upsilon \geq 0$ signifies the *balancing price*, which is the cost incurred to purchase the remaining quantity $Q^{b,H+1}$ outside of the PDA at the final time step $H + 1$.

At each round h , when a joint action $a^h \in A$ is executed, the state s^h transitions to the subsequent state $s^{h+1} = \{\mathcal{L}^{h+1}, \mathcal{Q}^{h+1}\}$, where \mathcal{L}^{h+1} represents the uncleared asks at round $h+1$ and \mathcal{Q}^{h+1} reflects the updated demands of the buyers after considering the cleared quantities from round h . Additionally, the sequence of transitions $s^1, a^1, s^2, a^2, \dots, s^H, a^H, s^{H+1}$, which starts at s^1 and ends at s^{H+1} , is denoted as a trajectory τ of the Markov game. The

trajectory τ generates a corresponding sequence of costs $C^{b,1}, \dots, C^{b,H}, C^{b,H+1}$ for each player $b \in [B]$.

At each visited state s^h , player b selects an action $a^{b,h} \in A^b$. The collection of actions chosen by player b at each round h is denoted by a Markov policy $\pi^b = \{\pi^{b,h} : S \rightarrow A^b \mid h \in [H]\}$. The joint policy $\pi = (\pi^b, \pi^{-b})$ includes player b 's policy π^b along with the policies of players except b , denoted as π^{-b} . The policy space for player b is represented by Π^b , while the combined policy space for all players is $\Pi = \times_{b \in [B]} \Pi^b$. To capture the cost of procurement, we define a value function $V_\pi^h : S \rightarrow \mathbb{R}$ for a joint policy $\pi \in \Pi$ at round h , as formalized in Equation (7.1).

$$V_\pi^h(s) = \sum_{r=h}^{H+1} C^{b,r}(s^r, a^{b,r}, a^{-b,r}). \quad (7.1)$$

More specifically, the value function is given by

$$V_\pi^h(s) = \sum_{r=h}^H \lambda^r \cdot \sum_{i \in [B^{b,r}]} c q_i^{b,r} + \Upsilon \cdot Q^{b,H+1}.$$

For the game \mathcal{G} , we define MPNE [135] using the value function in Definition 7.3.

Definition 7.3.1: Markov Perfect Nash Equilibrium (MPNE)

Given the game \mathcal{G} with H horizon and B buyers, a joint policy $\pi_* = (\pi_*^b, \pi_*^{-b})$ is an MPNE if $\forall b \in [B], \forall s \in S, \forall h \in [H]$, and $\forall \pi^b : S \rightarrow A^b$, we have

$$V_{\pi_*^b, \pi_*^{-b}}^h(s) \leq V_{\pi^b, \pi_*^{-b}}^h(s). \quad (7.2)$$

Intuitively, Equation 7.2 says that the value function under MPNE should lead to a joint policy that achieves the lowest procurement cost for the buyer b . Having described the Markov game framework, we will proceed to develop MPNE solutions in the next section.

7.4 Equilibria of the Markov Game

We begin by analyzing the case where each buyer $b \in [B]$ is restricted to submitting a single bid. Initially, we assume that the aggregate supply from bulk sellers is sufficient to satisfy the total demand, such that $Q^{\mathcal{S},h} \geq Q^{\mathcal{D},h}$. This assumption simplifies the equilibrium analysis, as the supply can meet the demand without constraint. However, exploring the scenario where the supply falls short is equally important, leading to a distinct equilibrium outcome compared to the surplus supply condition. This situation is critical to consider, as sellers may not always have the capacity to generate enough supply to fulfil the anticipated demand, thereby influencing the strategic behaviour of both buyers and sellers in the auction.

7.4.1 Adequate Supply Case

Let us define several key entities to clarify the proposed MPNE policy. First, denote $Q^{\mathcal{D}-b,h} = Q^{\mathcal{D},h} - Q^{b,h}$ as the cumulative demand of all players excluding player $b \in [B]$ at round h . Next, let u_h be the smallest index of an ask in the sorted set \mathcal{L}^h such that the cumulative supply from the first u_h asks exceeds the quantity $Q^{\mathcal{D},h}$. Formally, this can be expressed as $u_h = \arg \min_i (Q^{\mathcal{D},h} < \sum_{m=1}^i q_m^h)$. Similarly, let v_h^b be an index of the sorted set \mathcal{L}^h such that the supply up to first v_h^b asks is greater than $Q^{\mathcal{D}-b,h}$. That is,

$$v_h^b = \arg \min_j \left(Q^{\mathcal{D}-b,h} < \sum_{m=1}^j q_m^h \right) \forall b \in [B]. \quad (7.3)$$

Finally, define an index of \mathcal{L}^h as $z_h = \max\{v_h^1, v_h^0\}$ and let ϕ^h be the player who bids at a price p_{z_h} , where v_h^0 is $u_h - (H - h)$ and ϕ^h is defined as,

$$\phi^h = \max\{1, \arg \max_b \{v_h^b \leq v_h^0\}\}. \quad (7.4)$$

Here, z_h is the index of the supply curve; a player ϕ^h places the bid price equal to the price at this index. More specifically, the index z_h represents the number of rounds available for any player if he/she were to follow the supply curve starting from round h up to index u_h

on the supply curve. Furthermore, the player ϕ^h is defined as the player with the least demand who has at most v_h^0 rounds. Moreover, the ϕ is set to index 1 if $v_h^1 > v_h^0$, which emphasizes that the player with the highest demand can bid price defined by index v_h^1 and follow the supply curve from the round h onwards.

MPNE Policy for Adequate Supply Case: The joint policy π_* directs player b to first verify whether the condition $b = \phi^h$ is met. If this condition holds, the player should place a bid with the buy bid price $p_*^{b,h} = p_{z_h}$ and the corresponding quantity $q_*^{b,h} = Q^{b,h}$. On the other hand, if $b \neq \phi^h$, the player should submit a bid with the maximum buy bid price $p_*^{b,h} = p_{\max}$ and quantity bid equal to its requirement. The MPNE policy $\pi_*^{b,h}$ is formally defined in Equation (7.5).

$$\pi_*^{b,h} = \begin{cases} (p_{z_h}, Q^{b,h}) & \text{if } b = \phi^h \\ (p_{\max}, Q^{b,h}) & \text{o.w.} \end{cases} \quad (7.5)$$

Having explained the candidate policy π_* , the value function of the joint policy $\pi_*^{b,h}$ at round h for player $b \in [B]$ at state $s^h \in S$ is given in Equation (7.6) using the properties of the uniform clearing mechanism. More specifically, the clearing price at round h for all players is given by p_{z_h} and players $b \neq \phi^h$ are fully cleared, and player $b = \phi$ is partially cleared.

$$V_{\pi_*^b, \pi_{-b}}^h(s) = \sum_{r=h}^H \lambda^r \cdot c q^{b,r} + \Upsilon \cdot Q^{b,H+1},$$

$$= \begin{cases} \left[p_{z_h} \cdot Q^h - p_{z_h} \cdot \sum_{i \in [B] \setminus \phi^h} q_j^{i,h} + \sum_{k=h+1}^H p_{z_k} \cdot Q^k \right], & \text{if } b = \phi^h \\ p_{z_h} \cdot Q^{b,h}, & \text{o.w.} \end{cases} \quad (7.6)$$

Equilibrium Analysis

We establish that the candidate policy presented in Equation 7.5 indeed qualifies as the MPNE policy within the space of all deterministic policies. Specifically, we demonstrate that for every player $b \in [B]$, for all state $s^h \in S$, during any round $h \in [H]$, and under any deterministic policy π^b , the condition specified in Equation (7.2) holds true.

To substantiate this claim, we examine the value function across all potential deviations, with the possible bid deviations systematically categorized in Table 7.1. It's important to note that, by our assumptions, buyers are restricted from purchasing more than their required quantities; thus, only five distinct buy bid deviations are feasible. The combined deviations $\pi^b \in \Pi^b$ are defined as the Cartesian product of these bid deviations.

To build towards our main result, we first present a preliminary finding in Lemma 7.1. The first part of Lemma 7.1 provides a property of the clearing price λ^h for a uniform pricing clearing rule. The second part of Lemma 7.1 introduces a condition related to the cost of balancing outside the PDA's operational horizon.

Table 7.1: Possible Bid Deviations

Equal Priced Deviation	
$p^{b,h} = p_*^{b,h}, q^{b,h} < q_*^{b,h}$	
Higher Priced Deviations	Lower Priced Deviations
$p^{b,h} > p_*^{b,h}, q^{b,h} < q_*^{b,h}$	$p^{b,h} < p_*^{b,h}, q^{b,h} < q_*^{b,h}$
$p^{b,h} > p_*^{b,h}, q^{b,h} = q_*^{b,h}$	$p^{b,h} < p_*^{b,h}, q^{b,h} = q_*^{b,h}$

Lemma 7.1. *Given that the supply curve from bulk sellers across rounds $h \in [H]$ is constant and the clearing mechanism satisfies the properties of uniform clearing, we have the following.*

1. *The clearing price at h satisfy $\lambda^h \geq \lambda^{h+1}$ for $h \in [H-1]$.*
2. *The condition on the balancing price to buy outside the auction at $H+1$ denoted by Υ is given by $\Upsilon > \gamma \cdot p_{\max}$, where $\gamma > 1$.*

Proof. **Proof of statement 1:** Recollect that the clearing price λ^h is unique and belongs to an interval $[\hat{p}_d^h; p_+^{b,h}]$, where \hat{p}_d^h is the last cleared ask (or sell bid) at round h . Since the ask (or sell bid) is either partially or fully cleared, the lower limit of the interval would stay the same (in case of partially cleared) or increase (in case when fully cleared).

Proof of statement 2: Here, for any deviated policy π^b by the player b when all other players play Nash policy π_*^{-b} , the value function has to satisfy (7.7) for π_* to be an MPNE.

$$\begin{aligned} V_{\pi_*^b, \pi_*^{-b}}^h(s) &\leq V_{\pi^b, \pi_*^{-b}}^h(s) \\ V_{\pi_*^b, \pi_*^{-b}}^h(s) - V_{\pi^b, \pi_*^{-b}}^h(s) &\leq 0 \\ \sum_{r=h}^H (\alpha_*^{b,r} \lambda_*^r - \alpha^{b,r} \lambda^r) - \Upsilon Q^{b,H+1} &\leq 0 \end{aligned} \tag{7.7}$$

Here, only high-priced deviations or the combination of high-priced and low-priced deviations can lead to $Q^{b,H+1} = 0$. Hence, the value of Υ does not matter in this case. In addition, in case of higher-priced deviations alone, we can show that $\sum_{r=h}^H (\alpha_*^{b,r} \lambda_*^r - \alpha^{b,r} \lambda^r) \leq 0$.

Proof. Now, with the combination of low-priced and equal-priced deviations, the remaining quantity is $Q^{b,H+1} > 0$. In this case, the *worst case* condition on Υ is such that $\Upsilon > \frac{Q_{\max} \cdot p_{\max}}{q_{\min}} + p_{\max}$ which can be written as $\Upsilon > \gamma \cdot p_{\max}$, where Q_{\max} is the maximum quantity that is needed by any player and q_{\min} is the resolution (minimum sold quantity) of the auction market. Here, for all practical cases the range of Q_{\max} is lesser than the range of quantity q_{\min} , hence γ is a reasonable finite number. \square

We now move to the Theorem 7.1, which shows that the value function of the deviated policies is greater than the candidate policy's value function.

Theorem 7.1. *Given the conditions in Lemma 7.1 hold, we have that the bid deviations $\pi^b \in \Pi^b$ of player b satisfy MPNE condition shown in Definition 7.3.*

Proof. We first show the bid deviations yield higher costs compared to the candidate MPNE policy at $h = H$. To this end, for equal priced deviations, the cleared buy bid quantity for player b is $\alpha^{b,H} \leq \alpha_*^{b,H}$. Hence the value function^a for this deviation is

$$V_\pi^h = \lambda^H \cdot \alpha^{b,H} + \Upsilon \cdot Q_+^{b,H+1} \geq V_{\pi_*}^H$$

Next, for higher priced deviations (this deviation is not allowed when the player is recommended bidding at p_{\max}), the clearing price $\lambda^H \geq \lambda_*^H$ would increase, thereby increasing the value function. That is,

$$V_\pi^H = \lambda^H \cdot \alpha^{b,H} \geq V_{\pi_*}^H$$

^aThe state is suppressed in value function notation

Proof. Finally, for lower priced deviations, the cleared quantity might lower with a lower clearing price. However, it might lead to buying non-zero quantity outside the auction at $H + 1$, which is very expensive by Statement 2 of Lemma 7.1. Hence the value function is

$$V_\pi^H \geq V_{\pi_*}^H$$

Now assume at round $h = O + 1$ the value function satisfies $V_\pi^{O+1} \geq V_{\pi_*}^{O+1}$ for all $s \in S$.

We aim to show $V_\pi^O \geq V_{\pi_*}^O$ at $h = O$. To this end, for the equal price deviation at $h = O$ the cleared quantity decreases, that is $\alpha^{b,O} \leq \alpha_*^{b,O}$. This implies that the cost at round O decreases compared to candidate policy as

$$\lambda^O \cdot (\alpha^{b,O} - \beta^{b,O}) \leq \lambda_*^O \cdot (\alpha_*^{b,O} - \beta_*^{b,O})$$

However, using Statement ^a 1 of Lemma 7.1 and letting $\delta = \alpha_*^{b,O} - \alpha^{b,O}$, we have value function at O as

$$V_\pi^O = \lambda^O \cdot (\alpha^{b,O} - \beta^{b,O}) + \lambda^r \cdot \delta + V_\pi^{O+1} \geq V_{\pi_*}^O$$

, where $r > O$

Next, the higher priced deviations (when the player is not bidding p_{\max}) at round O will have a cost

$$\lambda^O \cdot (\alpha^{b,O} - \beta^{b,O}) \geq \lambda_*^O \cdot (\alpha_*^{b,O} - \beta_*^{b,O})$$

Hence by using $V_\pi^{O+1} \geq V_{\pi_*}^{O+1}$ and the immediate cost for higher priced deviation we have

$$V_\pi^O \geq V_{\pi_*}^O$$

^aThe Statement 1 of Lemma 7.1 says $\lambda^r \geq \lambda^O$, where $r > O$.

Proof. Finally, the lower priced deviations will have a cost $\lambda^O \cdot (\alpha^{b,O} - \beta^{b,O}) \leq \lambda_*^O \cdot (\alpha_*^{b,O} - \beta_*^{b,O})$ because of lower cleared quantity $\alpha^{b,h} \leq \alpha_*^{b,h}$ and lower clearing price $\lambda^h \leq \lambda_*^h$.

Here, using the condition on balancing price Υ and clearing price λ^r , $r > O$ from 7.1 we have that buying the quantities later will be either costlier or remain the same. That is, for this deviation the value function is

$$V_\pi^O = \lambda^O \cdot (\alpha^{b,O} - \beta^{b,O}) + \delta \cdot \lambda^r + V_\pi^{O+1} \geq V_{\pi_*}^O$$

□

7.4.2 Inadequate Supply Case

We now consider the inadequate supply, (that is, $Q^{\mathcal{S},h} < Q^{\mathcal{D},h}$). To provide the MPNE for this case, we modify the definitions of u_h , ϕ^h , v_h^k and z_h provided in the adequate supply case. First, we assign $u_h = |\mathcal{L}^h|$ as the value. Second, ϕ^h is given as

$$\phi^h = \arg \min_j \{Q^{\mathcal{S},h} \leq \sum_{b=1}^j Q^{b,h}\} \quad (7.8)$$

Third, when $\phi^h = B$, we set index v_h^N as

$$v_h^N = \arg \min_j \left(Q^{\mathcal{D}-B,h} \leq \sum_{m=1}^j q_m^h \right).$$

Finally, z_h when $\phi^h = B$ is defined as $z_h = \max\{v_h^\phi, v_h^0\}$. However, if $\phi^h < B$, then $p_{z_h} = p_{\max}$.

MPNE Policy for Inadequate Supply Case: The buy bids mentioned in Equation (7.5) now use modified definitions, whereas the sell bids change to bid at a price $p_-^{b,h} = \Upsilon - \epsilon$ and bid quantity $q_-^{b,h} = Q_-^{b,h}$. Similar to the adequate supply case, we provide the value function of the inadequate supply in Equation (7.9).

$$V_{\pi_*^b, \pi_*^{-b}}^h(s) = \begin{cases} p_{z_h} \cdot \left(Q^h - \sum_{i \in [N] \setminus \phi^h} q_j^{i,h} \right) + \sum_{k=h+1}^H p_{z_k} \cdot Q^k \\ \quad + \Upsilon \cdot Q_+^{b,H+1} & \text{if } b = \phi^h = N \\ p_{z_h} \cdot Q^{b,h}, & b < \phi^h \\ p_{z_h} \cdot \alpha_*^{b,h} + \Upsilon \cdot Q_+^{b,H+1} & b = \phi^h, \phi^h < N \\ \Upsilon \cdot Q_+^{b,H+1} & b > \phi^h \end{cases} \quad (7.9)$$

Note that for player $b > \phi^h$ equation $Q_+^{b,H+1} = Q^{b,h}$ holds.

In this analysis, we derived analytical equilibrium solutions for PDAs using a Markov game framework, operating under a *complete information setup*. In this scenario, buyers possess full knowledge of the supply curve and are aware of the demand information of other buyers during each round of the PDA. This comprehensive understanding allows for precise equilibrium calculations. However, real-world markets rarely afford such transparency. Therefore, in the next section, we extend our work by leveraging the equilibrium solutions presented in Equation (7.5) to develop a robust bidding strategy, MPNE-BBS, tailored for the more challenging *incomplete information case*. In this context, a buyer does not have access to the demand requirements of other players, nor does it have full insight into the supply curve.

7.4.3 MPNE-BBS Algorithm

We introduce a new algorithm, MPNE-BBS, specifically designed for the incomplete information environment of smart grids, drawing on insights from equilibrium analysis in complete information settings. The MPNE-BBS bidding algorithm is inspired by certain design principles from VidyutVanika21 (VV21) [29], making it important to first outline VV21's approach included in Chapter 6.

VV21 constructs the cost supply curve of GenCos by utilizing uncleared ask data provided by the simulator. The core idea is to determine the price that corresponds to the

buyer's desired bid quantity on the supply curve, which requires accurate forecasting of both the buyer's and the overall market's demands. This price is used as the upper limit for placing limit prices, with multiple bids positioned below this threshold. The rationale for placing multiple bids within this range is to maximize the chances of fulfilling most of the required quantity from the MISO buyer and other market players while using GenCo as a supplier of last resort. Since GenCo's supply curve is subject to random fluctuations between PDA rounds, placing multiple bids within a bounded range allows the agent to acquire energy at a lower cost.

Building on the strategy of VV21, MPNE-BBS identifies the optimal price on the supply curve by forecasting both the total market demand and the specific demand of the buyer, which are used to determine the indices u_h and v_h^b on the list of uncleared asks. These indices indicate the number of uncleared asks required to meet the demand, thereby helping to estimate the most advantageous limit price, consistent with the analytical solution presented in Section 7.4. After determining the price, MPNE-BBS places multiple bids similar to VV21 to account for the inherent variability of the supply curve. However, unlike VV21, MPNE-BBS focuses on directly acquiring the majority of the quantity from GenCo, thus ensuring a more reliable and cost-effective procurement strategy.

As outlined in the algorithm, MPNE-BBS starts by taking the current bidding timeslot as input and utilises the 24th hour (the first available opportunity) of each auction to observe the uncleared asks. Based on this observation, it generates a set of bids for each of the subsequent 23 hours. For each of these future timeslots (futureTime), the algorithm retrieves the list of uncleared asks from previous auction data, organized as a series of price and quantity pairs (p_i, q_i) , sorted in ascending order by price. Using this data, the algorithm estimates the bid price (estBidPrice) by leveraging market insights and its own demand forecasts.

MPNE-BBS

Algorithm 15: MPNE_BBS(*currentTime*)

Input: Current time *currentTime*

Output: list of bids *bidList*

```

1: totalDmd[] ← netDmdPredict(currentTime)
2:  $Q^{b,h}[] \leftarrow \text{indvDmdPredictor}[\text{currentTime}]$ 
3: for hour in  $[1, \dots, 23]$  do
4:   futureTime ← currentTime + hour
5:   unclearedAsks[] ← Auction(currentTime-1,futureTime)
6:   if unclearedAsks is not empty then
7:      $Q^{\mathcal{S},h} \leftarrow \text{sum}(\text{unclearedAsks.q})$ 
8:      $Q^{\mathcal{D},h} \leftarrow \text{totalDmd}[f\text{utureTime}]$ 
9:     if  $Q^{\mathcal{S},h} \geq Q^{\mathcal{D},h}$  then
10:       $p_{u_h} \leftarrow \min p_r \text{ s.t } Q^{\mathcal{D},h} \leq \sum_{i=1}^r \text{unclearedAsks.q}_i$ 
11:       $Q^{\mathcal{D}_{-b},h} \leftarrow Q^{\mathcal{D},h} - Q^{b,h}[\text{futureTime}]$ 
12:       $p_{v_h^b} \leftarrow \min p_r \text{ s.t } Q^{\mathcal{D}_{-b},h} \leq \sum_{i=1}^r \text{unclearedAsks.q}_i$ 
13:    else
14:       $p_{u_h} \leftarrow p_{|\mathcal{L}^h|}, p_{v_h^b} \leftarrow P_{|\mathcal{L}^h|}$ 
15:    end if
16:     $v_h^0 \leftarrow \max\{1, u_h - \text{hour} + 1\}$ 
17:    estBidPrice =  $\max\{p_{v_h^0}, p_{v_h^b}\}$ 
18:  else
19:    clearedPrices[] ← Auction( $t, t+\text{hour}$ )  $\forall t < \text{currentTime}$ 
20:    estBidPrice =  $\max\{\text{cleredPrices}\}$ 
21:  end if
22:  if (currentTime is far from futureTime) then
23:    minP =  $\beta_f * \text{estBidPrice}$ ; maxP =  $\delta_f * \text{estBidPrice}$ 
24:  else
25:    minP =  $\beta_c * \text{estBidPrice}$ ; maxP =  $\delta_c * \text{estBidPrice}$ 
26:  end if
27:  Sample  $P$  prices,  $p_i^{b,h} \sim U[\text{minP}, \text{maxP}]$ 
28:  Distribute  $Q^{b,h}$  uniformly across  $P$  prices,  $q_i^{b,h} \leftarrow Q^{b,h}/P$ 
29:  bidList ←  $(p_i^{b,h}, q_i^{b,h}) \forall i \in \{1, 2, \dots, P\}$ 203
30:  Auction(currentTime,futureTime) ← bidList
31: end for

```

The procedure for estimating the bid price, detailed in lines 4 to 21 of Algorithm 15, is derived from the equilibrium solution in Equation (7.5). Recognizing that the buyer has 24 opportunities to meet its demand, the algorithm allows for taking risks during the initial rounds while adopting a more conservative strategy in the final rounds, as outlined in lines 23 to 27. The hyperparameters β_f , δ_f , β_c , and δ_c can be fine-tuned to align with the buyer's risk tolerance. After uniformly sampling P prices, the buyer's required quantity is evenly distributed across P bids, which are then submitted for clearing.

7.5 Experimental Evaluation

In this section, we evaluate the performance of the proposed MPNE-BBS strategy against benchmark and state-of-the-art bidding strategies using the PowerTAC simulator. To ensure a comprehensive and rigorous comparison, we conduct experiments across a variety of market demand configurations. We begin by detailing the experimental setup, followed by an in-depth analysis and discussion of the results. Specifically, we compare MPNE-BBS against previously introduced strategies such as E-DDPGBBS. However, we do not include an extensive comparison between MPNE-BBS and SUPPLYCURVEBBS because the MPNE-BBS strategy is partially inspired by SUPPLYCURVEBBS. In certain special cases, both strategies result in identical bidding prices, making a direct comparison redundant. Finally, we analyze the proposed strategies, highlighting their respective strengths and weaknesses to provide a clearer understanding of their performance and applicability in dynamic market settings.

Benchmark: We have included details about the benchmark strategies in Section 4.2.2.

7.5.1 Experimental Setup

To comprehensively evaluate the efficacy of the proposed strategies, we conduct experiments under four distinct market demand levels: **Low**, **Med**, **High**, and **Extreme**. Since

E-DDPGBBS is a learning-based strategy whose performance heavily depends on training data influenced by market demand, we consider two experimental scenarios to assess its robustness.

In the first scenario, we train the E-DDPGBBS strategy on a specific market demand configuration and evaluate its performance across all demand levels. In the second scenario, we train separate E-DDPGBBS models for each market demand level and select the best-performing model for the corresponding configuration.

For each scenario, the experimental setup includes two variations: one with the inclusion of the MISO buyer and the other without it. Performance is measured in terms of unit purchase costs incurred by buyers in the wholesale market, where lower costs signify better performance. Each variation is further tested across the four market demand levels (**Low**, **Med**, **High**, and **Extreme**) using over 10 full-length PowerTAC games per configuration for greater statistical reliability.

We benchmark the proposed strategies against five established PDA bidding strategies—SPOT, VV18, VV21, ZI, and ZIP—as detailed in Section 4.4.2. The experiments are conducted under two distinct player configurations: 7-Player and 2-Player games. In 7-Player games, all available strategies are included, whereas 2-Player games involve games between E-DDPGBBS and MPNE-BBS. These experiments provide a comprehensive evaluation of strategy performance across a broad spectrum of scenarios.

Experimental Set-up for Scenario 1: In Scenario 1, we train the E-DDPGBBS strategy on a shared model for the **Low** and **Med** market demand configurations, with the MISO buyer participating in the training games. This trained E-DDPGBBS model is then applied across all experiments within this scenario.

- Set-1 involves 7-Player games, where each bidding strategy competes as an individual buyer alongside the MISO buyer. To ensure a fair comparison, all players (excluding MISO) must satisfy a fixed demand for each timeslot. Set-1 is further divided into

four configurations based on market demand levels: **Low**, **Med**, **High**, and **Extreme** (see Figure 7.1).

- Set-2 replicates the experiments from Set-1 but excludes the MISO buyer while maintaining the same four market demand configurations (see Figure 7.2).

Following the 7-Player games, we conduct 2-Player games to directly compare MPNE-BBS and E-DDPGBBS strategies under the same configurations. These experiments are performed for both Set-1 (with MISO) and Set-2 (without MISO) across all market demand levels (see Tables 7.4 and 7.5).

It is important to note that the E-DDPGBBS strategy in this scenario is not specifically trained for individual market demand levels, whether or not the MISO buyer is included. This allows us to evaluate the generalization capability of the E-DDPGBBS model across varying market conditions.

Experimental Set-up for Scenario 2: In Scenario 2, we extend our analysis by introducing additional E-DDPGBBS models trained specifically for varying market demand configurations. Building upon the previously trained E-DDPGBBS model for **Low** to **Med** market demand levels with the MISO buyer (denoted as *ED-Model1*), we now include three newly trained models tailored to distinct scenarios:

- *ED-Model2*: Trained for **High** to **Extreme** market demand with the MISO buyer participating in the games
- *ED-Model3*: Trained for **Low** to **Med** market demand without the MISO buyer
- *ED-Model4*: Trained for **High** to **Extreme** market demand without the MISO buyer.

During the analysis, we use the following models based on the market demand levels and the presence of the MISO buyer:

- For **Low** and **Med** demand levels: Use *ED-Model1* if the MISO buyer is present, and *ED-Model3* if the MISO buyer is absent.
- For **High** and **Extreme** demand levels: Use *ED-Model2* if the MISO buyer is present, and *ED-Model4* if the MISO buyer is absent.

The experimental setup for 7-Player games in Set-1 and Set-2 remains identical to that of Scenario 1 (refer to Figures 7.3 and 7.4).

In addition to this, we conduct 2-Player games between MPNE-BBS and each of the other strategies. Specifically, we play 10 games between MPNE-BBS and VV21, 10 games between MPNE-BBS and ZI, and so on, for all four demand levels across both sets (see Tables 7.4 and 7.5).

Furthermore, utilizing the E-DDPGBBS models trained for specific market demand configurations, we perform 2-Player games between E-DDPGBBS and MPNE-BBS across both sets and all four market demand levels, using the appropriate E-DDPGBBS models for each configuration (see Tables 7.3a and 7.3b).

7.5.2 Results

In this section, we discuss the results of the above-mentioned experiments, starting with the results of experiments under Scenario 1.

Experimental Results of Scenario 1: As illustrated in Figure 7.1, MPNE-BBS ranks among the top-performing bidding strategies in terms of wholesale cost, achieving either the best or second-best costs among seven brokers across various demand levels. Notably, at the **Extreme** demand level, MPNE-BBS's wholesale cost is just 6% higher than the lowest cost in the market and closely trails the second-best cost at **High** and **Med** demand levels (4.7% and 3.4% higher, respectively), with only VV21 outperforming it. The results are even more pronounced in Set 2, as shown in Figure 7.2, where MPNE-BBS consistently achieves a wholesale cost near the best in the market across all demand levels, outperforming VV21.

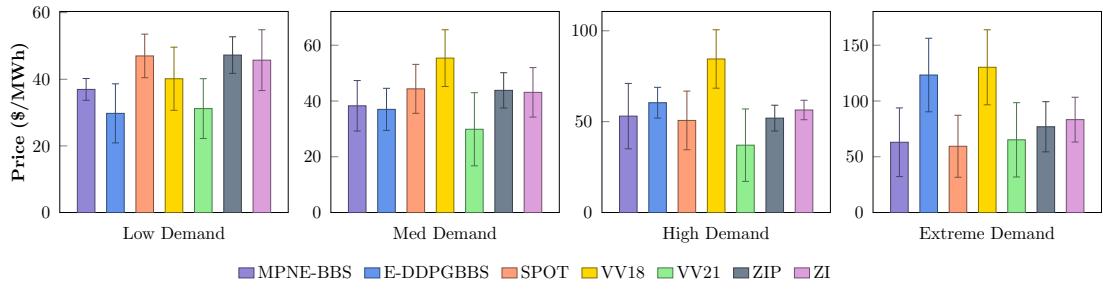


Figure 7.1: Scenario-1: Wholesale Cost Comparison in 7-Player Games (Set-1: with MISO Buyer)

In contrast, the performance of E-DDPGBBS varies significantly depending on the market configuration relative to its training data. The E-DDPGBBS model for Scenario 1 was trained for **Low** and **Med** market demand with the MISO buyer. As depicted in Figure 7.1, E-DDPGBBS performs as the best strategy in **Low** market demand and the second-best in **Med** market demand, even outperforming MPNE-BBS in these configurations. However, in other configurations, E-DDPGBBS exhibits high wholesale costs and often ranks as the worst-performing strategy, particularly in Set 2 (without the MISO buyer).

The results from the 2-Player experiments, shown in Tables 7.2a and 7.2b, compare the wholesale costs of E-DDPGBBS relative to MPNE-BBS. A value greater than 1 indicates that MPNE-BBS achieves a lower wholesale cost compared to E-DDPGBBS, and vice versa. The tables reveal that MPNE-BBS outperforms E-DDPGBBS in all configurations except **Low** and **Med** demand with the MISO buyer due to the same reason as explained above. Overall, these findings highlight MPNE-BBS's superior performance across most demand levels, whereas E-DDPGBBS excels in the configurations for which it was specifically trained but struggles with others.

Below, we discuss the results of experiments under Scenario 2 and show the performance E-DDPGBBS when trained for specific market demand configuration.

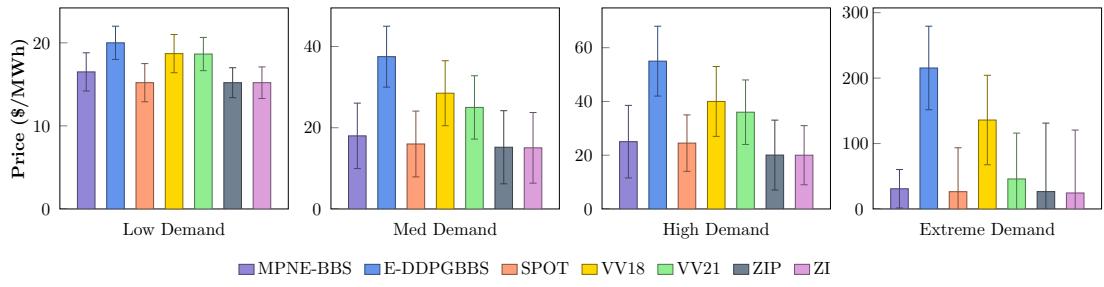


Figure 7.2: Scenario-1: Wholesale Cost Comparison in 7-Player Games (Set-2: without MISO Buyer)

Table 7.2: Scenario-1: Comparison of MPNE-BBS vs E-DDPGBBS in 2-Player Games

Opponent	Low	Med	High	Ext.	Opponent	Low	Med	High	Ext.
	E-DDPGBBS	0.73	0.85	1.12	1.64	E-DDPGBBS	1.08	1.07	1.02

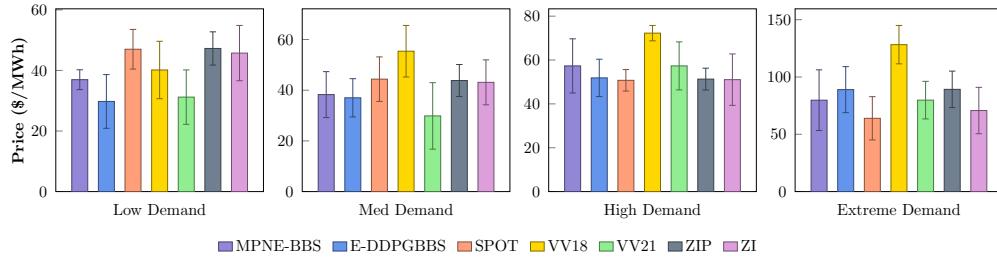


Figure 7.3: Scenario-2: Wholesale Cost Comparison in 7-Player Games (Set-1: with MISO Buyer)

Experimental Results of Scenario 2: Figures 7.3 and 7.4 illustrate that MPNE-BBS remains among the top-performing bidding strategies in terms of wholesale cost across varying demand levels in both sets of experiments. Notably, the major finding is the significant performance improvement of E-DDPGBBS when using configuration-specific

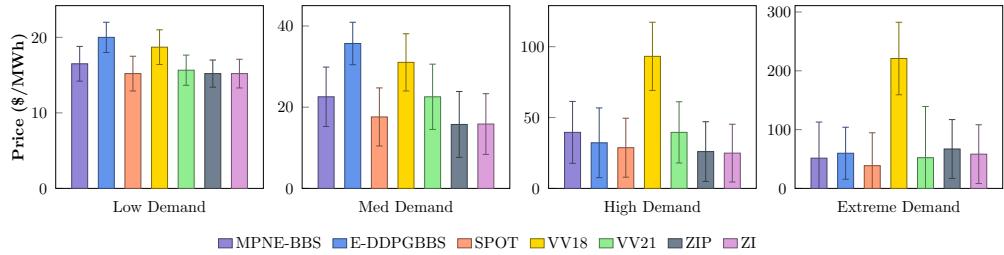


Figure 7.4: Scenario-2: Wholesale Cost Comparison in 7-Player Games (Set-2: without MISO Buyer)

Table 7.3: Scenario-2: Comparison of MPNE-BBS vs E-DDPGBBS in 2-Player Games

(a) Set-1: with MISO Buyer

(b) Set-2: without MISO Buyer

Opponent	Low	Med	High	Ext.	Opponent	Low	Med	High	Ext.
E-DDPGBBS	0.73	0.85	0.88	0.87	E-DDPGBBS	1.01	0.96	1.03	0.98

models. As shown in Figure 7.3, with the MISO buyer present, E-DDPGBBS emerges as the best or second-best strategy in all market demand configurations, except for the **Extreme** demand level, where it still performs competitively against MPNE-BBS and VV21. Although, in the absence of the MISO buyer, E-DDPGBBS does not surpass MPNE-BBS in all configurations, its performance shows a notable improvement compared to Scenario 1. Specifically, E-DDPGBBS ranks among the top strategies for **High** and **Extreme** market demand configurations and closely matches the wholesale costs of other strategies in the **Low** market demand configuration, as shown in Figure 7.4.

The results from the 2-Player experiments, detailed in Tables 7.3a and 7.3b, compare the wholesale costs of E-DDPGBBS relative to MPNE-BBS. As in Scenario 1, a value greater than 1 indicates that MPNE-BBS achieves a lower wholesale cost compared to E-DDPGBBS, and vice versa. The tables reveal that, except for a few configurations (**Low** and **High** demand in Set 2), E-DDPGBBS outperforms MPNE-BBS in most

Table 7.4: Performance in 2-Player Games (with MISO Buyer)

Opponent	Low	Mid	High	Extreme
VV21	1.01	0.98	1.03	1.18
DDPG	0.73	0.85	1.12	1.64
SPOT	1.45	1.09	0.87	0.77
VV18	1.32	1.36	1.49	1.76

scenarios. Even in cases where E-DDPGBBS trails behind MPNE-BBS, the difference between wholesale costs is minimal. These results highlight E-DDPGBBS’s improved performance and its emergence as a top strategy when trained specifically for the given market configurations.

In summary, the simulation results underscore MPNE-BBS’s adaptability and its ability to achieve significant performance improvements across various market demand scenarios and player configurations. Meanwhile, E-DDPGBBS proves to be a highly effective strategy, matching or exceeding MPNE-BBS in most 7-Player configurations and outperforming it in 2-Player configurations, particularly when trained for specific market demand configurations.

The results of the 2-Player experiments in Tables 7.4 and 7.5 show the wholesale cost of the opponent strategy relative to MPNE-BBS, a value more than 1 would indicate MPNE-BBS is the superior bidding strategy among the two. Particularly, it matches VV21, which is the best strategy in the literature and achieves similar wholesale cost as VV21 in all demand levels, with and without MISO buyer. It even outperforms VV21 for the extreme demand level where VV21’s wholesale cost is 1.18 times higher than MPNE-BBS’s cost. Overall, the results show that it achieves superior performance against each opponent and for almost all demand levels. This set of experiments aims to validate the

Table 7.5: Performance in 2-Player Games (without MISO Buyer)

Opponent	Low	Mid	High	Extreme
VV21	0.99	0.99	1.0	0.99
DDPG	1.08	1.07	1.02	1.08
SPOT	0.91	0.82	1.08	1.09
VV18	1.04	1.15	1.16	1.13

efficacy of MPNE-BBS against several different state-of-the-art strategies having different bidding patterns. Furthermore, we performed the same experiments for 3-Player and 5-Player games. On average, the relative costs of best and worst opponent strategies were 1.13 and 1.87, respectively, in 5-player without MISO games; 1.0 and 2.34, respectively, in 5-player with MISO games. Similarly, for 3-player games, these numbers are 1.27 and 1.47, respectively, for best and worst opponent strategies without MISO; 0.98 and 1.19, respectively, with MISO. Thus, the simulation results show that the MPNE-BBS achieves significant performance improvements against the best state-of-the-art bidding strategy across various numbers of players in a game for various market and demand scenarios.

7.5.2.1 Performance Analysis of E-DDPGBBS and MPNE-BBS

Based on the experimental results, we compare the strengths and weaknesses of E-DDPGBBS and MPNE-BBS across the two analyzed scenarios. In Scenario 1, the E-DDPGBBS agent, trained on the **Low** to **Med** demand configurations with the MISO buyer, was evaluated across all experiments, including varying demand levels and the presence or absence of the MISO buyer. The corresponding results are presented in Figure 7.1, Figure 7.2, Table 7.4, and Table 7.5. In Scenario 2, different E-DDPGBBS agents were trained for specific market demand levels and game configurations, as described in Sec-

tion 7.5.1. The results for this scenario are shown in Figure 7.3, Figure 7.4, Table 7.3a, and Table 7.3b.

From Scenario 1, it is evident that MPNE-BBS consistently outperforms E-DDPGBBS across most configurations. However, in **Low** and **Med** demand levels with the MISO buyer, E-DDPGBBS demonstrates competitive performance (Figure 7.1 and Table 7.4). This is expected, as the E-DDPGBBS agent was specifically trained on these configurations and struggles to generalize to other demand levels or game setups. In contrast, Scenario 2 demonstrates a substantial improvement in E-DDPGBBS’s performance. With configuration-specific E-DDPGBBS agents, E-DDPGBBS frequently surpasses MPNE-BBS in both 7-Player and 2-Player games, especially when the MISO buyer is present. This highlights the effectiveness of training E-DDPGBBS for targeted scenarios, allowing it to adapt and optimize effectively for specific demand configurations.

- **Adaptability:** MPNE-BBS demonstrates consistent and reliable performance across diverse market configurations, showcasing its adaptability and robustness in settings with limited information. In contrast, E-DDPGBBS’s performance is highly dependent on the training data and fails to generalize effectively in configurations for which it was not trained.
- **Configuration-Specific Training:** When E-DDPGBBS agents are tailored to specific demand levels and game setups, they outperform or match MPNE-BBS and other strategies, emphasizing the power of learning-based approaches when sufficient training data is available.

In summary, while MPNE-BBS delivers reliable, adaptable performance across diverse and dynamic configurations, a well-trained E-DDPGBBS agent can achieve superior results in targeted scenarios. This underscores the importance of configuration-specific training for learning-based strategies to maximize their performance in complex auction environments.

7.6 Summary

In this paper, we proposed equilibrium strategies for prosumers involved in a PDA to buy and sell commodities. We modelled the PDA as a Markov game with prosumers as players and derived equilibrium solutions for the complete information setting when the players are aware of the supply curve and the outstanding demand requirement of other players at every round of the PDA. Specifically, we derived MPNE solutions for the setting when the players compete to procure required commodities. Thereafter, the proposed MPNE solutions were used to design a bidding algorithm called MPNE-BBS for a more practical setup and its efficacy was demonstrated using the PowerTAC simulator test-bed against several state-of-the-art algorithms, specifically against E-DDPGBBS. The findings highlight that E-DDPGBBS excels with appropriate training for specific demand configurations, while MPNE-BBS offers adaptability across diverse scenarios. This underscores the efficacy of both strategies in enhancing PDA bidding.

Chapter 8

Designing Bidding Strategies using MCTS in Continuous Action Space

As discussed in the previous chapters, bidding in a periodic double auction (PDA) is challenging due to its **sequential nature**, where one needs to consider current as well as future auctions to decide the bids. We have previously seen a DDPG-based bidding strategy, which is offline in nature, meaning it requires training on simulated games before being deployed in real games. Then, our supply-curve based strategy, which is designed based on a few heuristic observations and MPNE-BBS does not involve any learning methods. **Monte-Carlo Tree Search (MCTS)**, which is a state-of-the-art **online planning algorithm** for tackling sequential problems, seems a perfect fit for bidding in PDAs where online decision-making is involved. However, the success stories of MCTS are largely limited to discrete action spaces, and its efficacy diminishes when dealing with continuous actions. Conventional methods often resort to overly simplistic discretizations that limit exploration and fail to provide valuable insights into unexplored actions. In this

work, we propose a novel bidding strategy for PDAs, **Regression-MCTS**, that is built upon MCTS for a **continuous action space of bid prices**. Unlike conventional methods, our novel MCTS method leverages information obtained from explored actions to enhance the understanding of the larger action set within the continuous domain to place bids in the auctions, thus generalizing the information about action quality between a wider action space for faster learning. To test the efficacy of our proposed method, we design an efficient PDA simulator that closely resembles real-world PDAs. Our analysis verifies that the increase in the number of rollouts improves its performance. Furthermore, our experimental results demonstrate that our approach outperforms existing MCTS-based bidding strategies and the majority of state-of-the-art PDA bidding strategies, showcasing its superior performance in PDAs¹.

8.1 Introduction

As discussed in the previous chapters, the design of optimal bidding strategies for PDAs is challenging, particularly in complex domains such as the energy market. In the energy market, the participants can procure energy several hours ahead of the delivery hour, necessitating strategic planning across current and future auctions, with each decision influencing subsequent steps. For such **sequential decision-making problems**, Monte Carlo Tree Search (MCTS) emerges as a fitting framework capable of constructing trees that encompass entire decision-making trajectories, comprehensively capturing process dynamics. Renowned for its efficacy in sequential decision-making tasks, MCTS gained widespread recognition after its pivotal role in AlphaGo’s triumph over the world champion in Go [136]. However, MCTS’s application predates its Go breakthrough, primarily in **game-playing**.

¹This work was published in the EMAS 2024 workshop, which was co-located with AAMAS 2024.

Numerous variations aiming to enhance search efficiency and reduce computational overhead have since been proposed. MCTS’s advantage lies in its ability to *blend the precision of tree search with the generality of random sampling*. Nevertheless, its performance in continuous spaces remains a subject of exploration, particularly in real-world problems with continuous action spaces, like determining velocity and acceleration in autonomous driving or setting bid prices in auctions. In these contexts, sequential decision-making is paramount, with each action step affecting the problem state in the future. While MCTS appears well-suited for such tasks, its adaptation to continuous action spaces requires further investigation and refinement.

In the literature, addressing the challenge of applying MCTS to continuous action space problems involves several approaches, including (i) *Discretization*, (ii) *Unpruning* or *Progressive Widening*, (iii) *Policy Optimization*, and (iv) *Action Optimization*. Discretization simplifies the continuous action space by discretizing it into a finite set of actions. However, this method constrains MCTS to operate within a fixed action set, limiting its ability to explore all potential outcomes and failing to provide insights into unexplored actions. Progressive Widening addresses the fixed action set limitation by dynamically expanding the action space with more and more simulations. Nevertheless, it also faces constraints due to discretization, restricting thorough exploration of potential outcomes. These techniques share a common limitation: they do not leverage insights gained from explored actions to inform the exploration of unexplored actions or enhance knowledge about previously explored ones.

Given the impracticality of exhaustively exploring all available actions in continuous space, any MCTS algorithm operating in this domain must *generalize the action space* based on exploration. Policy optimization and action optimization exhibit this generalization capability. Specifically, action optimization (techniques like KR-UCT [83]) aims to enable insights across the entire continuous action space with each action MCTS sample using the environment knowledge, whereas policy optimization methods take a hybrid

approach where they build the MCTS tree in a continuous action space and update the policy gradient using the sampled MCTS trajectories to train a parametric model; thereby outperforming discretization and progressive widening methodologies. However, both the above methods are curated for specific problem settings and may not be directly extendable to other problem settings.

In this study, we delve into the realm of continuous action spaces to explore the potential of MCTS, presenting a novel bidding strategy tailored for PDAs. Our strategy, named ***Regression-MCTS (R-MCTS)***, harnesses MCTS to derive optimal bidding prices from the continuous action space of prices. Unlike conventional approaches that limit exploration to discrete sets of candidate actions, R-MCTS considers the entirety of the continuous action space, leveraging a predefined set of candidate actions as initialization. Inspired by the KR-UCT method, the core of our strategy lies in generalizing action value estimates across the entire parameter space, facilitating information sharing and enabling exploration beyond the initial candidate set. This adaptability proves invaluable, especially in scenarios with ***imperfect domain knowledge***.

Central to our approach is the generalization of action value estimates over the bid price parameter space for PDAs, achieved by assigning a ***clearing probability*** to each action in the action space at each level of the Monte-Carlo tree, reflecting the ***likelihood of bid clearance***. As simulations progress, each exploration updates the clearing probabilities for all previously considered actions at any given level, enhancing MCTS's knowledge incrementally with each iteration. To evaluate the effectiveness of our method, we develop a ***comprehensive PDA simulation*** that closely replicates real-world dynamics. This simulation serves as a rigorous testing ground, enabling a comparative analysis of our proposed strategy against state-of-the-art MCTS approaches and top-performing PDA bidding strategies.

8.2 Regression-MCTS (R-MCTS)

In this section, we present our proposed method, R-MCTS, which operates within a *continuous action space* by leveraging *domain knowledge* to *generalize explored actions*. Our algorithm draws inspiration from KR-UCT’s [83] concept of facilitating information sharing among all considered actions, aiding in action space generalization and insights into unexplored actions (Refer explanation in Section 4.4.2). Additionally, to accommodate continuous action spaces, we employ the SPW technique (Refer Section 3.5.2). This involves initiating each node with a shallow action space and progressively expanding it as the node is visited repeatedly. Both initially provided candidate actions and progressively added actions, along with their estimated values, collectively form a single dataset at each node for R-MCTS.

Given the fundamental differences in the problem addressed in this work compared to previous approaches like KR-UCT, we do not model execution uncertainty for actions, as there is none in their execution of placing bids in a PDA. Consequently, we abstain from utilizing a kernel function to gain insights into similar unexplored actions. Notably, in the absence of execution uncertainty, KR-UCT essentially operates akin to the standard UCT algorithm with SPW, potentially limiting its efficacy in our problem domain. Instead, we utilize the knowledge of the market clearing of the bids to strengthen the insights about the actions that have already been visited; here, the actions are the bid prices. The core idea behind our approach is to calculate the clearing probability, $p_cleared$, for each of the explored actions (bid prices) and keep updating these probabilities with new information about market clearing.

Algorithm 16 delineates the pseudocode for R-MCTS, adhering to the conventional four-step process of MCTS: selection, expansion, simulation, and backpropagation. The primary procedure, R-MCTS, is invoked on the root of the search tree for a fixed number of rollouts determined by a computation budget, returning a list of bid prices for the buyer. The functions *generate_sellers()* and *clone_buyers()* emulate the sellers and buyers (including

R-MCTS and random buyers) of the PDA, respectively. Thus, these modelled sellers and buyers facilitate the simulation of PDA scenarios during the rollout process.

The modelled sellers primarily comprise GenCos, which follow a quadratic supply curve along with some noise. To model these GenCos, we get help from the PowerTAC simulator, which simulates such GenCos. The PowerTAC simulator generates GenCos ask with the help of pre-defined quadratic functions along with some nice to infuse stochasticity in the asks. We utilise the same set of quadratic functions along with noise in our simulator to model the GenCos. On the other hand, the modelled buyers consist of a clone of R-MCTS alongside ZI buyers that place bids randomly. The simulator also allows one to choose from a set of baseline buyers (see Section 8.4.2) instead of the ZI buyer to play against R-MCTS during the simulation. Throughout the rollout process, we replicate the actual market scenario using these modelled sellers and the same number of modelled buyers as in the original PDA, trading for demands identical to the original demands of buyers. Specifically, if in the actual market scenario, the R-MCTS is competing against n buyers where each of these buyers has their own demands; in the simulation, too, we replicate a similar scenario by cloning the same n number of competing buyers with their actual demand. The seller, too, would have similar asks and corresponding prices as to the actual market scenario. Since we may not have information about the opponents' actual bidding strategies, during the rollout, the ZI buyers or any other baseline buyers function as opponents of R-MCTS. This setup enables R-MCTS to assess the market scenario and make decisions accordingly. Further insights into the proposed algorithm are provided below.

R-MCTS

Algorithm 16: **R_MCTS**(*rem_quant*, *cur_ts*, *delv_ts*)

Input: remaining quantity *rem_quant*, current time *cur_ts*, delivery time *delv_ts*

Output: list of bids *bids*

```

1: bids  $\leftarrow \emptyset$ , root  $\leftarrow \text{Node}()$                                 # initialise bids list and root node
2: rem_auctions  $\leftarrow \text{delv\_ts} - \text{cur\_ts}$     # number of auctions remaining in a PDA
3: if rem_quant > 0 then
4:   while i in NUMBER_OF_ROLLOUTS do
5:     visited  $\leftarrow \emptyset$ , rewards  $\leftarrow \emptyset$ , cur  $\leftarrow \text{root}$ 
6:     visited  $\leftarrow [\text{visited}; \text{root}]$ 
7:     list_of_sellers  $\leftarrow \text{generate\_sellers}()$ 
8:     list_of_buyers  $\leftarrow \text{clone\_buyers}()$ 
9:     while not cur.is_leaf(rem_quant) do
10:    cur  $\leftarrow \text{select}(\text{rem\_quant})$                                # see algorithm 17
11:    cur.p_cleared  $\leftarrow \text{get_pcleared}(\text{rem\_auctions}, \text{cur.action})$ 
12:    cp, cq, rem_quant  $\leftarrow \text{perform\_auction}(\text{cur.action}, \text{rem\_quant},
13:      list_of_sellers, list_of_buyers)    # clearing the current auction round
14:    rewards  $\leftarrow [\text{rewards}; \text{cp}]$ 
15:    visited  $\leftarrow [\text{visited}; \text{cur}]$ 
16:   end while
17:   cur_cost, cur_quant  $\leftarrow \text{cur.simulation}(\text{rem\_quant}, \text{rem\_auctions})$ 
18:   root  $\leftarrow \text{backpropagation}(\text{rewards}, \text{visited}, \text{cur\_cost}, \text{cur\_quant})$ 
19:   end while
20:   lp  $\leftarrow \text{root.best\_action}()$ 
21:   bids  $\leftarrow [\text{bids}; \text{Bid}(\text{buyer\_ID}, \text{lp}, \text{rem\_quant})]$           # limit order
22: else
23:   bids  $\leftarrow [\text{bids}; \text{Bid}(\text{buyer\_ID}, \text{NULL}, \text{rem\_quant})]$           # market order
24: end if
25: return bids$ 
```

Select Method

Algorithm 17: select(*node*, *rem_quant*)

Input: Tree node *node*, remaining quantity *rem_quant*

Output: New state *new_state*

```

1: if number_of_visits(node)α ≤ number_of_children(node) then
2:   A ← actions considered in node
3:   action ← argmaxa ∈ A E(v|a) + C √ $\frac{\log \sum_{b \in A} \text{number\_of\_visits}(b)}{\text{number\_of\_visits}(a)}$  # UCB-select
4: else
5:   new action ← child of node by taking an action using p_cleared data
6:   node.children ← [node.children; action]
    # SPW-select: expanding action space
7: end if
8: new_state = node.action
9: return new_state

```

Calculation of Clearing Probabilities: The function *p_cleared*(*s, action*) is estimated from the past auction statistics as:

$$p_{\text{cleared}}(s, \text{action}) = \frac{\sum_{ac \in \text{auction}[s], ac.CP < \text{action}} ac.\text{cleared_amount}}{\sum_{ac \in \text{auction}[s]} ac.\text{cleared_amount}} \quad (8.1)$$

where *auction*[*s*] represents the collection of all past auctions in the current state *s*, with *CP* denoting the clearing prices of those auctions. Here, the state is defined by the number of auctions remaining in the PDA or the level in the Monte-Carlo tree, denoted as *rem_auctions* and computed based on current and delivery timeslots (Line 2). Essentially, this formula calculates the clearing probability of any *action*, which represents a bid price, by considering what fraction of the total quantity traded in the current state (*s*)

had a clearing price (CP) less than the given $action$. In simple terms, the formula calculates, historically, how much quantity has been traded at a price less than $action$ price, which indicates the probability or likelihood of a bid at $action$ price getting cleared. A higher price would have a $p_{cleared}$ value close to 1 as, historically, most of the quantity would have been sold at a comparatively lesser price and vice versa. In Algorithm 16, the $get_p_{cleared}(rem_auctions, cur.action)$ function executes the above-mentioned calculations for each action or bid price selected during the rollouts (Line 11).

8.2.1 Selection and Expansion Phase

The selection phase locates the leaf node in the current tree (the node that has not been expanded yet), and then we perform the expansion phase to expand the tree from that node (Lines 9-15). Our algorithm invokes the $select$ method (Line 10), which determines the next node to traverse by utilizing two selection modes: UCB-select and SPW-select, as depicted in Algorithm 17. UCB-select adheres to the UCB formula at each decision node, aiding the algorithm in selecting from previously explored actions. At each node, crucial metrics such as the number of visits, the action that led to the current node, the average unit purchase cost (mean utility $\mathbb{E}(v|a)$), and a set of child nodes b are maintained. As iterations progress and outcomes are revisited, their mean utilities and visit counts are updated accordingly. These updated estimates are then integrated into their respective roles within the UCB formula, guiding the selection of actions for further refinement. The scaling constant C plays a pivotal role, governing the tradeoff between exploring less-visited actions and refining the value of more promising actions.

SPW-select is triggered when the condition for SPW 3.5.2 is satisfied (Line 1), indicating that a node has been visited sufficiently many times compared to its number of children. This condition signifies the necessity to expand the action space for the current node. This decision is based on maintaining the number of outcomes in a node bounded by some exponential function of the number of visits to the node. To execute the SPW step, a

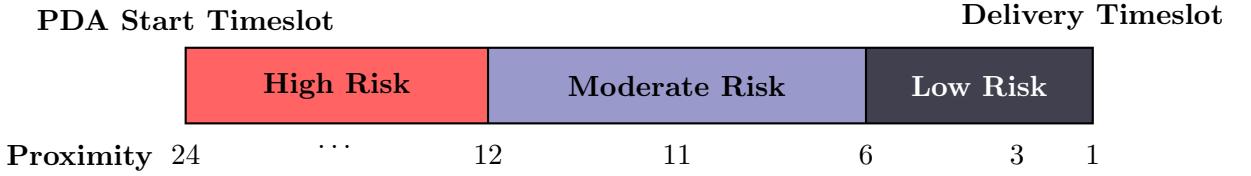


Figure 8.1: Tactics to Adjust Risk between Auction Rounds

random policy is employed to select an action beyond the initial action space. Alternatively, domain knowledge can be utilized to select a new action.

During the SPW stage, we leverage the *knowledge of $p_{cleared}$* data in the current state of the node to introduce a new action into the action space. Given that PDA allows a buyer to bid for 24 rounds per auction, our strategy varies based on risk tolerance. In early rounds, we opt for actions with lower $p_{cleared}$, taking higher risks to procure the required quantity at lower prices. As the auction progresses towards its final rounds, we adopt a more conservative approach, prioritizing actions with higher $p_{cleared}$ to ensure procurement certainty, as shown in Figure 8.1. This strategy enables R-MCTS to explore risky actions (or bid prices) initially and transition towards more assured options as the auction progresses. Once the new action is chosen, it is added to the list of children of the current node, and the function returns the next state after executing the selected action.

Once the algorithm reaches a leaf node in the tree, it expands the tree by selecting an action and conducting auction clearing (*perform_auction()*) for that chosen action. This process results in transitioning to the next state, which becomes a child of the current node. Subsequently, the simulation phase is conducted from this new state.

8.2.2 Simulation and Backpropagation Phase

When a new outcome is integrated into the tree, a complete search ensues to reach a terminal state using a default policy, which can be a random policy as well. We simulate the remaining rounds of the auction from the state generated during the expansion phase

and record the procurement cost along with the quantity purchased by R-MCTS (Line 16). During the simulation phase, at each of the remaining auction states (*rem_auctions*), we conduct auction clearing (*perform_auction()*) and record intermediate procurement costs. Additionally, we update the remaining quantity of all the brokers for the subsequent state based on the market clearing in the current state. Subsequently, all the intermediate procurement costs and cleared quantities of R-MCTS are aggregated and returned as the output of the simulation phase. This output is then utilized to update $\mathbb{E}(v|a)$ and the visit count of the visited nodes along the selected path through the tree during the backpropagation phase (Line 17).

8.2.3 Final Selection Phase

Once the computational budget is exhausted, determined either by the number of iterations or by a pre-allocated time limit, we utilize the search results to select an action for execution. The chosen action is the one at the root with the *highest mean utility*, indicating the lowest unit purchase cost.

Figure 8.2 shows an intermediate state of R-MCTS. Each node contains VC, act, MU, $|C|$ denoting the visit count of the node, action or limitprice that leads to the node, mean utility, and number of children, respectively. Below, we provide implementation details of R-MCTS with the help of the example given in Figure 8.2.

8.3 Example Walk-through of R-MCTS

In this section, we explain the working of the proposed Algorithm 16 in more detail with the help of an example shown in Figure 8.2. At the start, the tree is empty. The algorithm starts with initialising the *root* node of the tree and the list of *bids*. Then, it creates the search tree based on the current auction round (*rem_auctions*). After that, if the remaining quantity (electricity) to purchase for the broker is positive, then it starts the ROLLOUT routine; otherwise, it places a market order to see the quantity (if the remaining

quantity is negative, that means the broker has extra quantity and it needs to be sold). In the rollout routine, the algorithm performs *NUMBER_OF_ROLLOUTS* iterations; the number of iterations depends on the available computation budget.

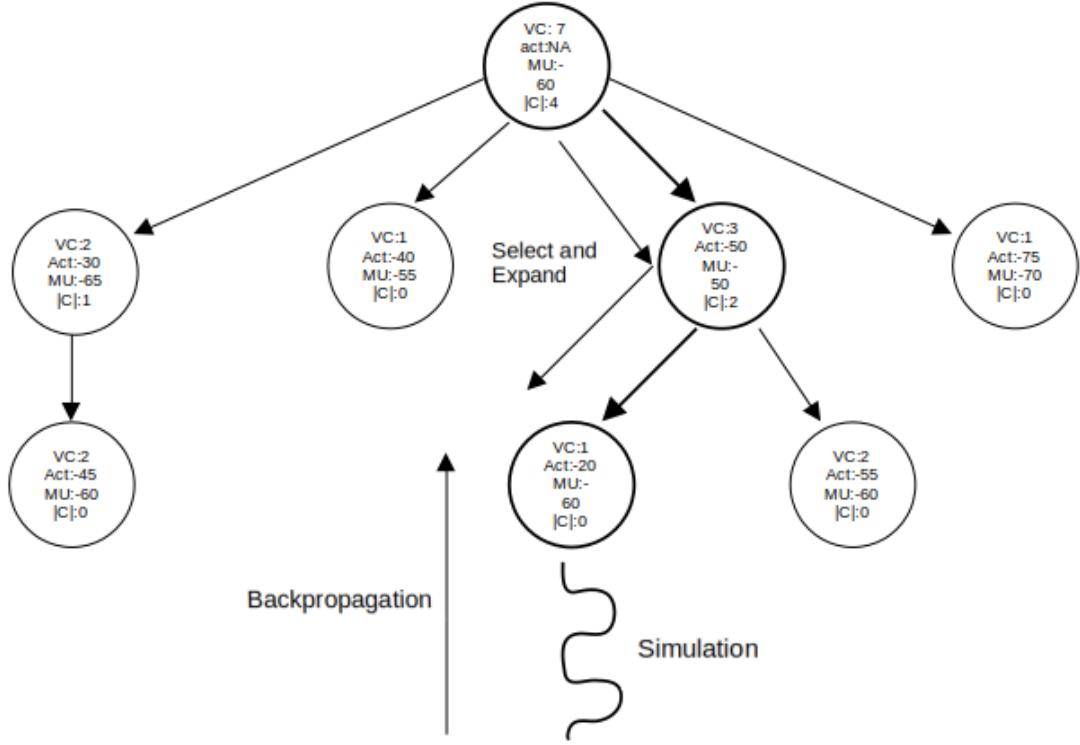


Figure 8.2: Pictorial Representation of R-MCTS Displaying all the Phases

During the rollout process, the algorithm initialises the essential data structures to store the list of visited nodes (*visited*) and rewards (*rewards*). Here, the *reward* is the average clearing price of PDA during each rollout process. A new node *cur* is created and assigned to the *root* node of the tree; this node helps the algorithm to traverse the search tree. The rollout routine starts with adding the *root* node in the *visited* list. Then, to simulate the rollout process, it generates the list of sellers and list of buyers as described in Section 8.2. After this, the algorithm follows the standard four-step process of MCTS: selection, expansion, simulation and backpropagation. First is selection, followed

by expansion, combined in the algorithm (line no. 9 to 15). Then, the simulation step (line no. 16) and the backpropagation step (line no. 17). This four-step process is repeated at each iteration of the rollout process.

In our example, in the first iteration of rollout, there is only a single node in the tree. Thus, the selection process would select that node and expand the tree from that selected node. In order to expand the tree, the algorithm selects an action (a bid price) and places a bid in the PDA. The auction clearing mechanism collects this bid along with bids and asks from the generated sellers and buyers. The mechanism determines the clearing price cp , clearing quantity cq and remaining quantity rem_quant for all the players in this auction. After this, a new node is added to the search tree and is added to the *visited* list. The cq is a reward and is added to the *rewards* list. The visit count for the new node becomes 1 (initially 0 and gets incremented by 1 after each visit. After the expansion step, the simulation step occurs, where the tree is traversed till all rounds of auction get exhausted. During each round, a PDA auction occurs exactly as discussed above, and the cp, cq and rem_quant are recorded. Finally, once the simulation is over, the reward is calculated as the average clearing price during the simulation using the data of cp and cq , which is then added to the *rewards* list. Finally, the backpropagation step takes place, and all the visited nodes get updates with their respective rewards; also, $p_cleared$ of each visited node gets updated based on auction data till the current iteration.

The above process is repeated multiple times to expand the tree in breadth and depth directions. Once the rollout process is over, the algorithm selects the *best_action* from the *root* node of the tree, which is the action that leads to the child node having the lowest average clearing price. Once the action is known, a bid is placed in the actual market PDA.

8.4 Experimental Evaluation

In this section, we conduct experiments to assess the performance of R-MCTS within continuous action spaces. As R-MCTS deals with a continuous action space, it can effec-

tively handle any number of actions by expanding the tree in breadth. Specifically, we apply R-MCTS to the domain of PDAs, where it generates bid prices from a continuous range of feasible prices. To facilitate our evaluation, we have developed an efficient PDA simulator that faithfully replicates real-world PDA dynamics. This simulator serves as our testing platform to thoroughly examine R-MCTS and compare it against baseline strategies. We begin by detailing the experimental setup, followed by descriptions of each of the opponent baseline strategies. Finally, we present results alongside discussions.

8.4.1 Experimental Setup

To thoroughly evaluate the proposed R-MCTS, we conduct a comprehensive analysis alongside state-of-the-art MCTS strategies and efficient PDA bidding strategies. Our experiments are organized into three sets. Set-1 focuses on examining the learning capabilities of R-MCTS by analyzing its performance relative to the number of rollouts. Specifically, we compare the average purchase cost of R-MCTS across varying rollout numbers to ascertain whether increased rollouts lead to improved performance. Set-2 involves comparing the purchase cost of R-MCTS against various other MCTS-based bidding strategies in 1v1 games (only two buyers in PDAs, R-MCTS and one of the opponent strategies) under different demand scenarios (*low*, *medium*, *high*, and *extreme*). By performing these experiments, we aim to demonstrate the efficacy of our proposed approach against state-of-the-art MCTS methods. Similarly, Set-3 entails a comparison of R-MCTS against potent bidding strategies from the PowerTAC literature in 1v1 games across all four demand scenarios. This experiment assesses the performance of our strategies against some of the best PDA bidding strategies.

By conducting these comparisons across different demand levels, we aim to test the efficacy of our strategies under diverse circumstances. Each experiment is executed 100 times to ensure robust results, with mean values presented in the results. Below, we provide

detailed explanations for each set of experiments.

Experiment Set-1: In this set of experiments, we engage R-MCTS as the sole participant in a PDA. Each iteration involves randomly generating demand for the strategy from a Gaussian distribution. To fulfil this demand, R-MCTS conducts a series of rollouts to familiarize itself with the environment and determine optimal bidding strategies. The purpose of these experiments is to assess the learning proficiency of R-MCTS across varying numbers of rollouts, ranging from low to high. Specifically, we explore the following rollout numbers: $\{1, 5, 10, 50, 100, 500, 1000\}$. For each rollout quantity, we record the unit purchase cost of R-MCTS, and the corresponding graph is depicted in Figure 8.3. This analysis provides insights into how the performance of R-MCTS evolves with increasing rollout numbers.

Experiment Set-2: This set of experiments aims to assess the performance of the R-MCTS bidding strategy against various other MCTS-based strategies. To achieve this, we leverage several state-of-the-art MCTS methods to design bidding strategies. Specifically, we implement bidding strategies based on Vanilla-MCTS and MCTS-SPW [87]. Additionally, we incorporate the SPOT [81] agent, renowned as one of the top-performing strategies in PowerTAC tournaments. Each of these strategies engages in 1v1 competition against R-MCTS. Following each individual bidding round, we record the average unit purchase costs of each opponent, including R-MCTS. These experiments are conducted across four different demand levels: (i) Low, (ii) Medium, (iii) High, and (iv) Extreme. R-MCTS's results are computed based on 500 rollouts. The resulting graph is presented in Figure 8.4.

Experiment Set-3: The third set of experiments mirrors Set-2, albeit with a broader scope. Here, we compare the R-MCTS bidding strategy against a comprehensive array of efficient bidding strategies drawn from the PowerTAC literature. These strategies embody

some of the most effective approaches for PDAs, encompassing diverse attributes such as learning-based strategies using Reinforcement Learning, Heuristics-based methods, and more. Specifically, we included ZI, ZIP, VV21, and SPOT strategies in our evaluation. Similarly to Set-2, each of these strategies engages in 1v1 competition against R-MCTS across all four demand levels. We meticulously record the average unit purchase costs of each opponent alongside R-MCTS following individual bidding rounds. R-MCTS’s results are derived from 500 rollouts. The comparative performance graph is displayed in Figure 8.5.

8.4.2 Baseline Strategies

Below, we provide concise descriptions of the baseline strategies employed as opponents in our experiments. These strategies are categorized into two groups: MCTS-based strategies and PowerTAC bidding strategies. Please refer to Section 4.4.2 for PowerTAC bidding strategies.

8.4.2.1 MCTS-based Strategies

MCTS-Vanilla: MCTS-Vanilla is an MCTS method utilizing a discrete and fixed-size action space, constructed iteratively through a search tree as detailed in Section 3.5.1. For designing a bidding strategy for PDAs, MCTS-Vanilla discretizes the action space of bid prices. Each action within this space is defined by two multipliers, α_1 and α_2 , applied to the lower and upper bounds of feasible bid prices, respectively. The resulting bid is then placed in the PDA, with the remaining quantity placed as bid quantity across all auction instances. Procurement cost, determined by the bid price, serves as a reward propagated back through the tree.

MCTS-SPW: MCTS-SPW (Simple Progressive Widening) extends MCTS-Vanilla to accommodate continuous action spaces better. While still maintaining a discrete action space, MCTS-SPW dynamically grows its size with the number of rollouts. In the context

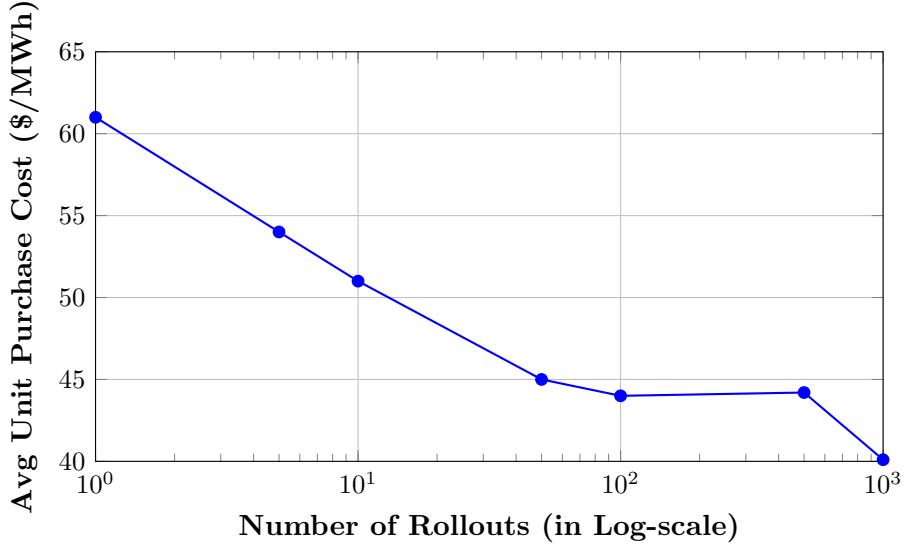


Figure 8.3: Rollouts vs Average Unit Purchase Cost for R-MCTS

of designing a bidding strategy for PDAs, MCTS-SPW initializes the action space with randomly sampled bid prices. With more and more rollouts, it explores actions outside the initial action space while striking a balance between exploration and exploitation. Like MCTS-Vanilla, it allocates the remaining quantity as a bid quantity across all auction instances.

8.4.3 Results and Discussion

Below, we present the results of each of the three experiments mentioned above, averaging over 100 random rollouts.

Experiment Set-1: Figure 8.3 depicts the results from Set-1 experiments, illustrating the impact of the number of rollouts on the average unit procurement cost of R-MCTS. As evident in the graph, there exists an inverse relationship between the number of rollouts and the procurement cost: as the number of rollouts increases, R-MCTS enhances its

performance and procures the demand at lower prices. Consequently, the efficacy depends on the number of rollouts feasible within the system. The result can also be viewed as an effect of the number of actions on the average unit procurement cost. The higher the number of rollouts, the broader the breadth of the Monte-Carlo tree, which would mean that more actions under any tree node can be explored. As results suggest, an increase in the number of rollouts leads to a decrease in the procurement cost, signifying the impact of exploring more and more actions in order to generalize the continuous action space.

Experiment Set-2: Figure 8.4 illustrates the results of Experiment Set-2, comprising three graphs that compare R-MCTS against three opponents across four different demand scenarios. Specifically, Figure 8.4(a) compares the R-MCTS against MCTS-Vanilla, Figure 8.4(b) against MCTS-SPW, and Figure 8.4(c) against SPOT. As depicted in Figures 8.4(a) and 8.4(b), R-MCTS consistently outperforms MCTS-Vanilla and MCTS-SPW across all demand levels—low, medium, high, and extreme. With the exception of extreme demand, R-MCTS achieves significantly lower procurement costs compared to MCTS-Vanilla and MCTS-SPW across all scenarios. Similarly, as evident in Figure 8.4(c), except for low-demand, R-MCTS matches SPOT’s performance in medium-demand scenarios and outperforms SPOT by a substantial margin in high and extreme-demand scenarios.

Experiment Set-3: Figure 8.5 presents the results of Experiment Set-3, comprising three graphs comparing R-MCTS against three opponents across four different demand scenarios. Specifically, Figure 8.5(a) compares R-MCTS against MCTS-ZI, Figure 8.5(b) against MCTS-ZIP, and Figure 8.5(c) against VV21. As depicted in Figure 8.5(a), R-MCTS consistently outperforms ZI across all four demand levels. Similarly, as shown in Figure 8.5(b), except for extreme demand scenarios, R-MCTS outperforms ZIP by a substantial margin and nearly matches its performance in extreme demand scenarios. A possible explanation for ZIP outperforming R-MCTS in an extreme demand scenario may be that ZIP main-

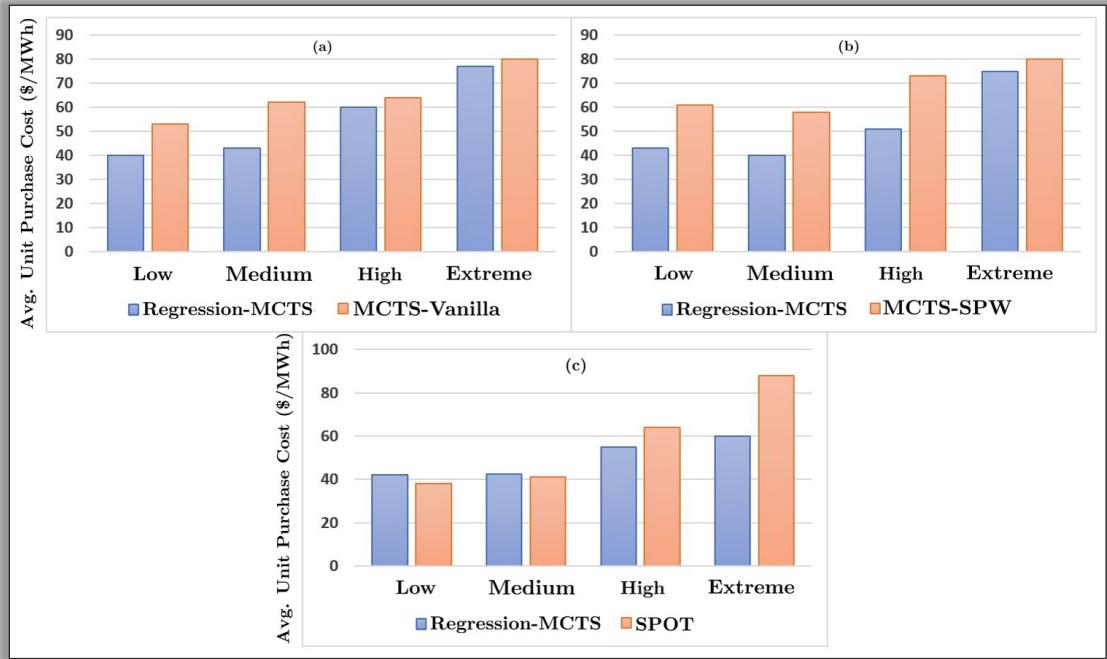


Figure 8.4: Comparing R-MCTS vs MCTS-based Bidding Strategies in Low, Med., High and Ext. Demand Scenarios (((a) vs MCTS-Vanilla (b) vs MCTS-SPW (c) vs SPOT))

tains a profit margin while placing bids, and its unit procurement costs show very low variance; however, R-MCTS has small but increasing variance as we move from low demand to extreme demand. Thus, ZIP's ability to maintain a steady procurement cost helps it to perform slightly better than R-MCTS. Finally, as illustrated in Figure 8.5(c), while VV21 outperforms R-MCTS in low and medium-demand scenarios, R-MCTS maintains its performance in high and extreme-demand scenarios, surpassing VV21 by a considerable margin. Notably, VV21 is regarded as the best strategy for PowerTAC PDA, and our proposed strategy proves to be more robust across different demand scenarios.

These experiments underscore the effectiveness of our proposed R-MCTS method in the continuous action space of bid prices for PDAs. It shows performance enhancement with an increasing number of rollouts. Furthermore, our method consistently outperforms sev-

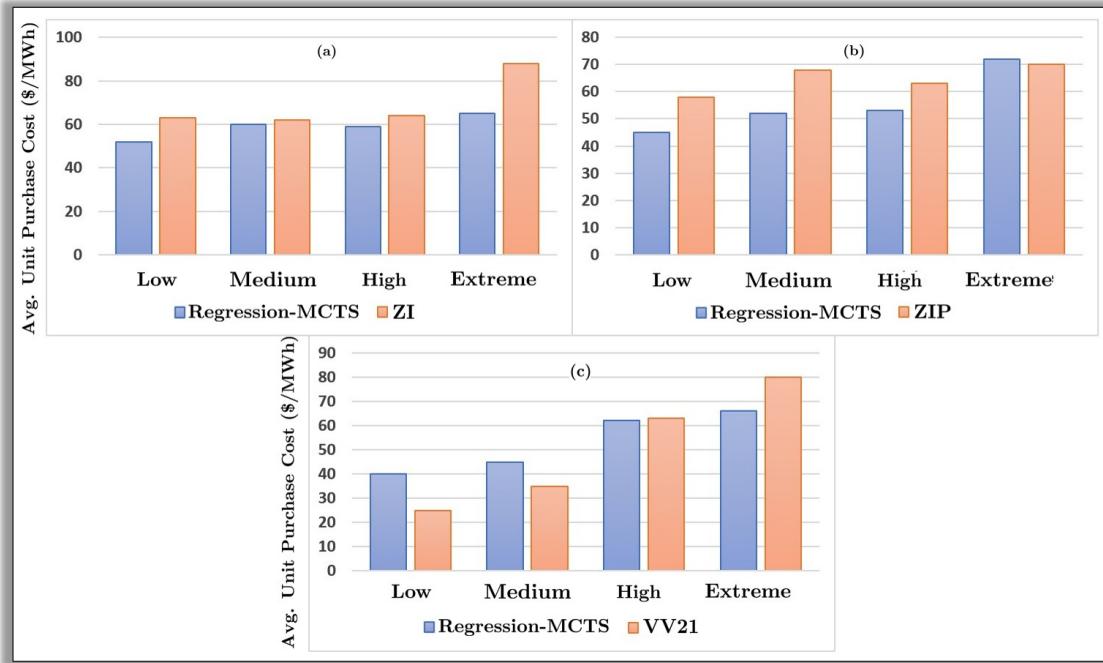


Figure 8.5: Comparing R-MCTS vs PowerTAC Bidding Strategies in Low, Medium, High and Extreme Demand Scenarios (((a) vs ZI (b) vs ZIP (c) vs VV21))

eral top-tier MCTS-based and PowerTAC PDA bidding strategies across different demand levels, underscoring its robustness in adapting to varying demand scenarios.

8.5 Summary

In this work, we delve into the underexplored realm of Monte Carlo Tree Search (MCTS) for continuous action spaces, presenting a novel bidding strategy named R-MCTS that extends MCTS to the continuous action space of bid prices, aiming to improve auction bidding strategies. By leveraging information from explored actions, R-MCTS enhances understanding of the larger action space within the continuous domain, facilitating more informed bidding decisions. Specifically, we utilize clearing probabilities of bid prices calculated based on MCTS’s previous auction bids, thereby generalizing information about action quality. To evaluate our method’s effectiveness, we developed a realistic PDA simulator closely resembling real-world scenarios. Our analysis reveals that increasing the number of MCTS rollouts enhances performance. Moreover, R-MCTS outperforms existing MCTS-based baseline bidding strategies and many state-of-the-art PDA bidding strategies, showcasing its superior performance in PDAs.

Part B: The Tariff Market

B₁: Designing Tariff Generation Strategies
in Tariff Market

Chapter 9

Tariff Generation via Heuristic Method

In the previous segment, we showcased our contributions to designing bidding strategies for the wholesale market. Another important aspect of the functioning of smart grids is designing **tariff-generation strategies**. The emergence of smart grid technology has opened the door for wide-scale automation in decision-making. A broker, an integral part of a smart grid system, has to procure electricity from the wholesale market and then sell it to customers in the retail market by publishing attractive tariff contracts. It can deploy autonomous agents to make decisions on its behalf. In this work, we describe the tariff contracts generation strategy of one such autonomous agent, **VidyutVanika21**. We call our proposed strategy **GenTariffs-Heur**. Supported by game-theoretic analysis, the GENTARIFFS-HEUR designs tariff contracts that a) maintain a balanced portfolio of different types of customers, b) sustain an appropriate level of market share, and c) introduce surcharges on customers to reduce energy usage during peak demand times¹.

¹This work has been published as a part of IJCAI 2022 paper.

9.1 Introduction

The brokers play a significant role in smart grid operations and are responsible for the efficient functioning of the system. The major tasks of brokers are to buy electricity from the wholesale market and sell it to retail customers by generating ***lucrative yet profitable tariff contracts*** and managing the supply-demand balance in the smart grid system.

As discussed in Chapter 4, the retail market of a smart grid incorporates various types of customers like households, office spaces, villages, producers (customers having solar panels or wind turbines), electric vehicles, battery storage, and a few others. Some of these customers have the capability to change their electricity usage pattern based on the signals from the broker, commonly in the form of tariff contracts. Tariff contracts can also be of multiple types to cater to a variety of customers. For example, (i) ***Fixed Price Tariff (FPT)*** having the same rate values for all hours in a day/week, (ii) ***Time of Use (ToU)*** tariff having different rate values for different hours in a day/week, (iii) ***Tier tariffs*** having different rate values corresponding to different usage slots, (iv) ***variable tariffs*** where rate values can change dynamically, or (v) combination of any of the above tariff types. The brokers decide the appropriate tariff types and tariff rate values for the customers in their portfolios.

The smart grid system is quite complex in nature, and it is practically impossible to test or validate the new strategies on the real-world smart grid system. Thus, we use PowerTAC as a testbed to test our proposed strategy. PowerTAC embodies a variety of customer models to represent the wide variety of customers as seen in the real world. It supports all kinds of tariff contracts mentioned earlier. Furthermore, PowerTAC introduces a balancing market that handles the real-time balancing of supply and demand. It penalizes agents in case of an imbalance in their portfolio.

In this work, we specifically focus on the ***tariff contract generation*** problem in the retail market of the smart grid. To generate a new tariff contract, an agent needs to decide the tariff contract type and the tariff contract's rate values. The tariff contracts are ***public***

information; any agent and a customer in the simulation can see all the active tariffs in the retail market. Thus, if an agent does not adapt to the changing market situation and does not update its tariff contracts periodically, any opponent agent can offer better tariff contracts and take away all the customers. Thus, it is paramount to update tariff contracts periodically, which can be done using either heuristic-based approaches or learning-based approaches.

As discussed in Section 4.4, the proposed strategies in the PowerTAC literature, except TUC-TAC, sought to maximize the revenue/profit without explicitly controlling the agent’s market share, which is not an optimal strategy due to capacity transaction penalties. Furthermore, most of the above retail strategies, including TUC-TAC, have been generic and are not effectively specialized for different player configurations. To overcome the above problems, we design a tariff strategy inspired by the game theory literature and decide the optimal market share for various player configurations. After this, it uses heuristic-based techniques to achieve and maintain that market share during the simulation. In addition, tariffs contain surcharges for peak hours, resulting in customers shifting loads to off-peak hours apart from providing a cushion to handle peak demand charges. We analyze the performance of GENTARIFFS-HEUR in the PowerTAC 2021 tournament and showcase its efficacy through the tournament results.

9.2 Market Share Analysis

This section presents the *game-theoretical analysis* to decide an *optimal market share* for various player configurations of PowerTAC games, which is then used in the GENTARIFFS-HEUR to design suitable ToU tariffs. We show the analysis for three different player configurations of PowerTAC games, namely, 2-Player, 3-Player, and 5-Player games. We construct a utility matrix for each player configuration by modelling the PowerTAC games as *two-player zero-sum games*, solving which results in an equilibrium market share. We are modelling the scenario as two-player games as all the other brokers, irre-

spective of the number of brokers, are the opponents of our broker, thus treating them as a single opponent. This way of modelling helps in deciding opponents' strategies based on the combinations of opponents, as their real strategies are unknown.

We utilize the analysis below to design a heuristics-based tariff strategy for our broker, VidyutVanika, during the PowerTAC'21 and PowerTAC'22 tournaments. The tariff strategy aimed to maintain the market share suggested by the game theory analysis using intelligent heuristics.

The **utility** of the row player is defined in Equation 9.1, which is the difference between the average final cash positions of the row and column players. This way of modelling the utility matrix helps us to maximize the difference between VidyutVanika's average cash position and the opponent's average cash position, thereby helping VidyutVanika generate higher profits than opponents. As we formulate this as a zero-sum game, the column player gets negative of the utility calculated in Equation 9.1.

$$u_1(s_i, s_{-i}) = \frac{1}{T} \sum_{i=1}^T x_i - \frac{1}{n} \sum_{k=1}^n \left(\frac{1}{T} \sum_{i=1}^T y_{ik} \right) \quad (9.1)$$

In Eq. 9.1, x_i denotes the final cash balance of VidyutVanika in the game i , while y_{ik} denotes the final cash balance of opponent agent k in the game i and n denotes the number of opponent agents in the game. For our analysis, the average values are taken over $T = 5$ games. Note that we play $T = 5$ games for each combination of VidyutVanika's strategy and opponents' strategy. For example, in Table 9.1, we play a total of $35 * 5 = 175$ games, with each game requiring approximately 2 hours to complete. Given the significant time constraints involved, we chose to run the experiments for 5 games per strategy combination as a reasonable compromise. This allows us to obtain good approximations of the outcomes while maintaining practical feasibility within the available time frame.

In our modelling, we select VidyutVanika as the row player, and a subset of opponents, depending on the player configuration, act as a sole column player. The row player's

Table 9.1: 2-Player Games Analysis (Utility Values in Millions)

Opp. VV	TT	VV18	VV20	A	C
0%	-1.12	-0.73	-1.41	-0.73	-0.63
15%	-2.92	-1.74	-2.03	0.32	-1.61
30%	-1.92	-0.88	-1.10	0.33	-0.34
45%	-0.51	-0.18	0.01	0.73	0.21
60%	-0.06	0.59	0.46	0.44	1.38
75%	-0.14	1.33	0.08	-0.93	1.88
100%	-0.27	1.00	-0.34	-0.85	0.78

(VidyutVanika's) strategy set is given by $S_1 = \{0\%, 15\%, 30\%, 45\%, 60\%, 75\%, 100\%\}$, where each element in the set S_1 specifies the target market share that VidyutVanika has to maintain during the simulated games. We have five agents from past PowerTAC tournaments to act as opponents in our analysis, namely, TUC_TAC (TT) [96], VidyutVanika18 (VV18) [28], VidyutVanika20 (VV20), CrocodileAgent (C) and AgentUDE (A) [110]. The column player strategy set S_2 depends on the player configuration. For example, in a 2-Player game configuration, we need only one opponent against VidyutVanika; thus, $S_2 = \{TT, VV18, VV20, C, A\}$. Similarly, in a 3-Player game configuration, we need two opponents against VidyutVanika, which is to be selected from the available set of five agents; thus, total $5c2 = 10$ elements is the set S_2 as shown in Figure 9.2.

Equilibrium Calculation: Figures 9.1, 9.2, 9.3 and 9.4 show the utility matrices for 2-Player, 3-Player, 5-Player and 7-Player configurations, respectively. Each cell describes the utility value, a cash difference in millions calculated by playing a set of T games. The same process is repeated for all the combinations of VidyutVanika's strategies (S_1) and

Table 9.2: 3-Player Games Analysis (Utility Values in Millions)

Opp. VV	TT, VV18	TT, VV20	VV18, VV20	TT, C	TT, A	VV18 , C	VV18 , A	VV20 , C	VV20 , A	C, A
0%	0.11	-1.21	-0.12	-0.69	-0.14	-0.90	0.36	-0.91	0.61	0.73
15%	0.23	-1.48	0.85	-0.85	0.12	-0.32	1.09	-0.45	0.63	0.65
30%	0.09	-0.40	0.98	-0.17	-0.21	-0.19	1.24	-0.02	0.87	0.85
45%	0.85	0.14	1.38	0.16	0.16	0.14	1.31	0.20	0.86	0.86
60%	1.07	0.18	1.37	0.04	0.20	0.56	1.03	-0.25	0.84	0.83
75%	0.96	-0.56	1.33	-0.17	0.14	-0.10	1.02	-0.17	0.33	0.18
100%	0.79	-1.01	0.47	-0.55	-0.51	-0.61	-0.06	-1.39	0.03	-0.07

Table 9.3: 5-Player Games Analysis (Utility Values in Millions)

Opp. VV	TT, VV18, VV20, C	TT, VV18, VV20, A	TT, VV18, A, C	TT, A, VV20, C	A, VV18, VV20, C
0%	-0.89	-0.30	-0.17	-0.16	1.74
15%	-0.20	-0.02	-0.20	-0.15	1.58
30%	0.11	-0.05	0.11	0.04	1.90
45%	-0.08	0.04	0.16	0.14	1.81
60%	-0.31	0.03	-0.29	-0.10	1.74
75%	-0.49	-0.23	-0.37	-0.41	1.02
100%	-0.50	-0.56	-0.19	-0.19	1.00

opponents' strategies (S_2) to create the full utility matrix. Thereafter, we use Gambit [137]

Table 9.4: 7-Player Games Analysis (Utility Values in Millions)

Opp.	VV	0%	15%	30%	45%	60%	75%	100%
All	0.1077	0.2034	0.3408	-0.1078	0.2333	-0.1149	0.061	

to solve the game and output the Nash Equilibrium. We found that each of the above three player configurations exhibits Mixed Strategy Nash Equilibrium (MSNE).

- **For 2-Player Configurations:** Based on Figure 9.1, the utility matrix leads to Pure Strategy Nash Equilibrium of 60% market shares.
- **For 3-Player Configurations:** Based on Table 9.2, the utility matrix leads to MSNE of randomizing between 45% and 60% market shares with probabilities 0.8 and 0.2, respectively, which results in equilibrium market share of 48% ($0.8 * 45 + 0.2 * 60$).
- **For 5-Player Configurations:** Based on Figure 9.3, the utility matrix leads to MSNE of randomizing between 30% and 45% market shares with probabilities 0.43 and 0.57, respectively, which translates to equilibrium market share of 38.55% ($0.43 * 30 + 0.57 * 45$).
- **For 7-Player Configurations:** In the 7-Player games, we introduce one more broker called Sample Broker provided by PowerTAC; all the available opponent brokers act as a single strategy, as shown in Table 9.4. We found that the above game has a dominant strategy of playing with a 30% market share strategy. Note that the strategy of playing with 60% market share also generated significant utility. Hence, we decided to aim for a larger market share than 30%.

The same results can be seen visually as well; the bold letters in the tables show the strategies having the higher utilities $u_1(\sigma_1^*, \sigma_{-1}^*)$ than the remaining strategies $u_1(\sigma_1, \sigma_{-1}^*)$ for row-player VidyutVanika, which leads to above-calculated MSNEs.

9.3 Design of GenTariffs-Heur

GENTARIFFS-HEUR handles tariff publication and revocation in the retail market. The strategy used by GENTARIFFS-HEUR in the retail market is inspired by game theory and aims to maintain an appropriate level of market share in the retail market. Furthermore, the retail module also aims to achieve a balanced portfolio of customers of different types (consumption, production and storage). Tariffs usually contain surcharges during peak hours to further mitigate the effect of capacity transaction charges. Our design of GENTARIFFS-HEUR, along with wholesale strategy, helps in handling balancing market activities as well. During high-demand scenarios, when GenCo cannot fulfil all of the market demand, GENTARIFFS-HEUR supplies energy via producers and storage customers in its portfolio to make up for the market deficit and, in the process, generates revenue.

GENTARIFFS-HEUR has three sub-modules, namely, *Tariff Enhancer (TE)*, *Tariff Designer (TD)*, and *Tariff Health Checker (THC)*. The TE sub-module calculates mean tariff rates to maintain retail market share within predefined bounds. The TD sub-module generates weekly ToU tariffs based on the mean rates suggested by TE with surcharges during peak hours. The THC sub-module periodically checks for active tariffs' profitability and subscription rates and removes loss-making and under-subscribed tariffs. Algorithm 18 shows the working of GENTARIFFS-HEUR. Figure 9.1 provides an overview of the proposed strategy. As shown in the figure, we maintain three hyperparameters HB, LB, and MB, which correspond to higher, lower, and middle bounds. The values of HB, MB and LB are mentioned in Section 9.4. Note that these values are decided based on the study of the historical PowerTAC games and offline evaluation of the strategy. Below, We detail TE, TD, and THC.

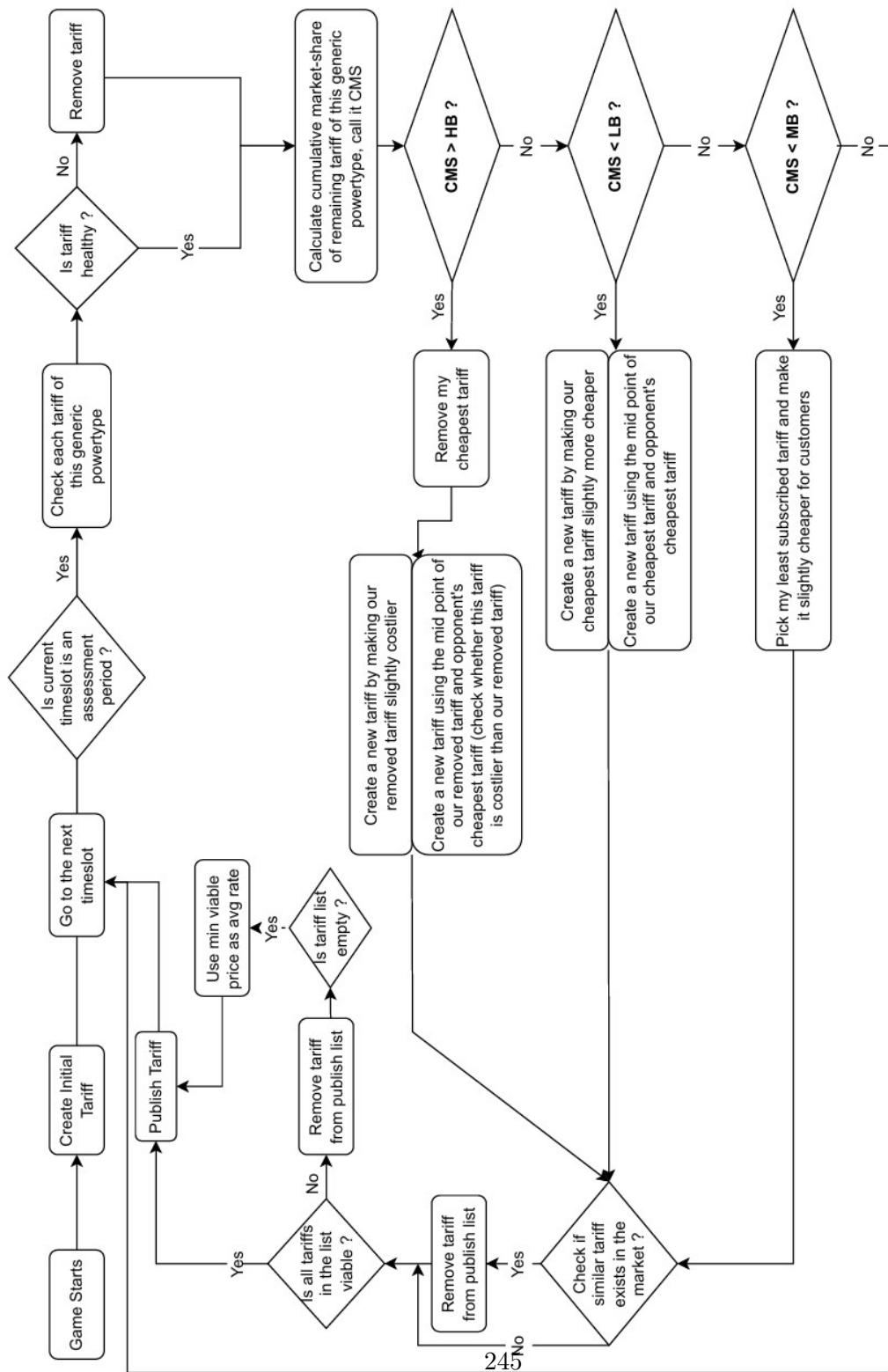


Figure 9.1: GEN TARIFFS-HEUR Flowchart

* slightly cheaper indicates: decrease avg rate for consumption tariff by 10% / increase avg rate of production tariff by 10%

* slightly costlier indicates: increase avg rate for consumption tariff by 10% / decrease avg rate of production tariff by 10%

* similar tariffs are the tariffs where the difference between avg rate of tariffs in within 10% of the avg rate of the new tariff

GenTariffs-Heur

Algorithm 18: GenTariffs_Heur()

Input: None

Output: List of active tariffs $activeTariffs$

```

1:  $activeTariffs \leftarrow \text{GenerateInitialTariffs}()$ 
2: if assesmentInterval then
3:    $revokedTariffs \leftarrow \text{TariffHealthChecker}(activeTariffs)$ 
4:    $\text{RevokeTariffsFromMarket}(revokedTariffs)$ 
5:    $activeTariffs \leftarrow activeTariffs \setminus revokedTariffs$ 
6:    $newTariffs \leftarrow \text{TariffEnhancer}(activeTariffs)$ 
7:    $\text{PublishTariffs}(newTariffs)$ 
8:    $activeTariffs \leftarrow activeTariffs \cup newTariffs$ 
9: return  $activeTariffs$ 
10: end if

```

9.3.1 The Tariff Enhancer (TE) Module

TE is in charge of maintaining an appropriate level of market share in a game by improving our broker's active tariffs by considering all active tariffs present in the market. Recall that aiming to monopolize the retail market will result in high penalties for peak-time usage. Therefore, a broker needs to carefully choose market share bounds and publish tariffs to maintain market share within the chosen bounds. More precisely, we maintain three hyperparameters **HB**, **LB**, and **MB**, which correspond to higher, lower, and middle bounds. Based on empirical evidence, TE aims to maintain retail market share between **MB** and **HB**, which helps it attain higher profits through sufficient retail market revenue without being penalized heavily for peak time usage. Based on the game-theoretical analysis, we choose values for these hyperparameters for each player configuration.

Algorithm 19 shows the working of TE, which suggests average tariff rates for all customer types (referred to as **powerType**). First, TE computes the cumulative market share

of all its active tariffs using the ***EstimateMarketShare()*** method for each customer type. Thereafter, if our broker's current market share is higher than HB, to reduce market share, TE removes the broker's cheapest tariff and suggests costlier tariff rates by calling the method ***GenerateCostlierTariffs()***. Similarly, if the market share is lower than LB, TE calls ***GenerateCheaperTariffs()*** that suggests cheaper tariff rates to help increase the market share. In the case when the market share is between LB and MB, TE calls ***FineTuneTariffs()*** to fine-tune the broker's cheapest tariff by making it slightly cheaper to achieve the desired market share bounds. TE checks the viability of new tariff rates to ensure that the mean tariff rates are higher than the average energy procurement costs before calling the TD sub-module to generate weekly ToU tariffs.

9.3.2 The Tariff Designer (TD) Module

TD is responsible for designing a weekly ToU tariff based on the average input price (***avgPrice***) received from TE as shown in Algorithm 20. First, TD calls ***DefineWeekly-TariffPattern()*** method that generates a binary weekly tariff pattern indicating peak and non-peak hours. This pattern is designed by analyzing the historical net market demand values retrieved from the logs of past PowerTAC tournaments. For example, we found that the morning and evening hours of a day have high demand, and these hours are marked as peak hours along with several other peak hours identified from the analysis. Then TD calls ***DefineSurplusMultipliers()*** method to define surplus multipliers s_i for each of the 168 hours in a week. These multipliers are greater than 1 for peak hours and are set to 1 for non-peak hours. Specifically, for peak hours, the value of s_i would depend on the magnitude of the peak as observed from market demand data. We empirically observed that such tariffs help in mitigating the effect of capacity transaction penalties.

Tariff Enhancer

Algorithm 19: Tariff Enhancer(*activeTariffs*)

Input: List of active tariffs *activeTariffs*

Output: New tariffs newToUTariffs

```
1: for powerType in offeredPowerTypes do
2:   tariffs  $\leftarrow$  getTariffs(activeTariffs, powerType)
3:   marketShare  $\leftarrow$  EstimateMarketShare(tariffs)
4:   if marketShare  $\geq$  HB then
5:     RevokeCheapestTariff()
6:     newTariffs  $\leftarrow$  GenerateCostlierTariffs(tariffs)
7:   else if marketShare  $\leq$  LB then
8:     newTariffs  $\leftarrow$  GenerateCheaperTariffs(tariffs)
9:   else if marketShare  $\leq$  MB then
10:    newTariffs  $\leftarrow$  FineTuneTariffs(tariffs)
11:  end if
12:  newToUTariffs = []
13:  for tariff in newTariffs do
14:    if isTariffViable(tariff) then
15:      ToUTariff  $\leftarrow$  TariffDesigner(tariff, powerType)
16:      newToUTariffs  $\leftarrow$  newToUTariffs  $\cup$  ToUTariff
17:    end if
18:  end for
19: end for
20: return newToUTariffs
```

Tariff Designer

Algorithm 20: Tariff Designer(*avgPrice*, *powerType*)

Input: Average price *avgPrice* and Power type *powerType*

Output: Time of use tariff *ToUTariff*

```
1: pattern  $\leftarrow$  DefineWeeklyTariffPattern().  
2: s[]  $\leftarrow$  DefineSurplusMultipliers(pattern)  
3: find normRate :  $\frac{\sum_{i=1}^{168} s_i * \text{normRate}}{168} = \text{avgPrice}$   
4: rate[i]  $\leftarrow s_i * \text{normRate}$ , for i  $\in \{1, 2, \dots, 168\}$   
5: ToUTariff  $\leftarrow$  CreateTariff(rate, powerType)  
6: return ToUTariff
```

9.3.3 The Tariff Health Checker (THC) Module

THC aims to remove unhealthy tariffs from the retail market periodically. Specifically, it checks the accounting information of all the active tariffs of our broker, using which it calculates the profit/loss of tariffs and its average subscription rate. The profits obtained by a tariff are calculated using revenue generated by the tariff from its subscribers, which results in lower energy procurement costs. Loss-making or under-subscribed tariffs are marked as unhealthy and removed from the market.

9.4 Deploying GenTariffs-Heur in PowerTAC

The analysis in Section 9.2 suggests how our broker should randomize in targeting market share. On top of this randomization, due to the stochasticity of the PowerTAC simulation and customer models, it is difficult to maintain one particular market share across different games. Hence, we aim to maintain market share within specific bounds such that the broker’s average market share is close to the equilibrium market share. Based on the above analysis, for PowerTAC 2021, we decided to keep HB around 45% for 7 and

5-player configurations and 60% for 3-player configurations. The values of MB and LB are set to 60% and 40% of the HB. Note that these values are decided based on the study of the historical PowerTAC games and offline evaluation of the strategy.

Tariff Health Checker

Algorithm 21: Tariff_Health_Checker(*tariffs*)

Input: Broker's active tariffs in the market *tariffs*

Output: Tariffs to be revoked *revoked*

```

1: Let revoked = [].
2: for t in tariffs do
3:   calculate profit p of t.
4:   calculate average subscription avgSub of t
5:   if p ≤ 0 or avgSub ∉ [tlb, thb] then
6:     add t to revoked.
7:   end if
8: end for
9: return revoked

```

PowerTAC 2021 finals analysis shows that our broker maintained an average market share of approximately 38%, 40%, and 55% in 7, 5, and 3 player configurations, respectively, which are in line with the equilibrium market shares. We also carried out an ablation study to showcase that GENTARIFFS-HEURstrategy of achieving equilibrium market share contributed immensely to the success of our broker during the PowerTAC2021 tournament. We presented an exhaustive analysis of the PowerTAC2021 tournament in Chapter 12, where we showcase the performance of GENTARIFFS-HEUR in the tournament.

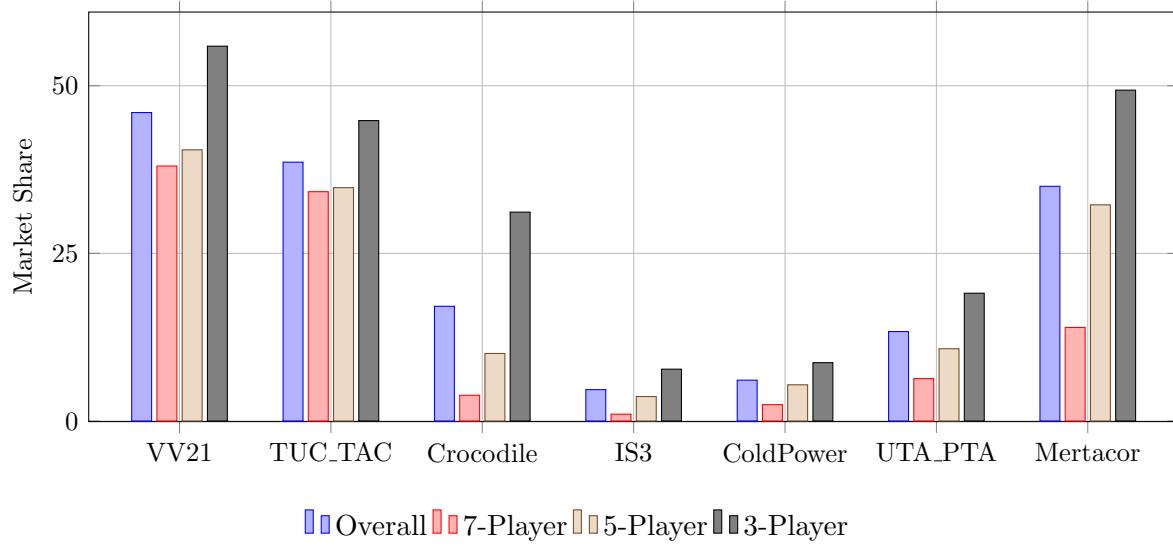


Figure 9.2: Average Market Share of Brokers in PowerTAC 2021

9.5 Experiments and Results

We deployed the proposed strategy in our agent, VV21, for the PowerTAC2021 tournament. As VV21’s tariff strategy focuses on maintaining an optimal market share for each player configuration, we analyze its average market share throughout the tournament, as shown in Figure 9.2. The optimal market shares, derived from game-theoretical analysis for 7-Player, 5-Player, and 3-Player configurations, are 30.0%, 38.55%, and 48%, respectively (see Section 9.2). During the PowerTAC2021 finals, VV21 maintained average market shares of 38.03%, 40.44%, and 55.9% in the 7-Player, 5-Player, and 3-Player configurations, respectively. These values are close to the optimal targets, demonstrating that VV21 effectively maintained its market share as expected during the finals.

9.6 Summary

In this chapter, we presented the details of the working of retail modules of our broker, GENTARIFFS-HEUR. The novelty of GENTARIFFS-HEUR lies in utilizing game theory to determine optimal retail market share bounds specific to player configuration. We presented a thorough analysis to determine an appropriate level of market share in all different player configurations. After determining the optimal market share, we designed a heuristics-based strategy to achieve and maintain the appropriate level of market share during the PowerTAC games. We described the strategy in detail with the use of a flowchart and explained how all its submodules work. Finally, we described the deployment of the GENTARIFFS-HEUR in the PowerTAC and presented some of the results showcasing its efficacy in maintaining an appropriate level of market share during the PowerTAC2021 tournament.

Chapter 10

Tariff Generation via Contextual-MAB Method

As discussed in the last chapter, a broker has to procure electricity from the wholesale market and then sell it to customers in the retail market by publishing attractive tariff contracts. In the last chapter, we discussed a heuristic-based tariff contracts generation strategy. In this chapter, we describe a **learning-based tariff contracts generation** strategy motivated by the **Contextual MAB** literature. We particularly utilize the **EXP-3** for Contextual MAB-based learning. We call our proposed strategy **GenTariffs-EXP3**. In the last chapter, we showed that maintaining an appropriate market share in the retail market yields high net revenue. Thus, taking the game-theoretic analysis that determines an optimal level of market share as a base, we propose a Contextual MAB-based strategy to achieve and maintain the suggested level of market share by adapting to the market situation and revising the tariff contracts periodically. We validate our proposed strategy in PowerTAC and showcase that it is able to maintain the suggested market share¹.

¹This work was published in the EMAS 2023 workshop, which was co-located with AAMAS 2023.

10.1 Introduction

In the last chapter, we discussed the need for a tariff generation strategy and the advantages of having an AI-based strategy that can make decisions on its own. We also discussed the types of customers and types of tariff contacts available in the PowerTAC simulator. Even though the heuristic-based strategy presented in the last chapter, GENTARIFFS-HEUR, is effective, there is still human expertise required to lay out a contingent plan for all possible scenarios that may happen during the smart grid operation. Thus, having a learning-based strategy that can learn the optimal response in various different scenarios by interacting with the environment is a promising approach.

As discussed in the previous chapter, smart grid technology enables the use of adaptive autonomous agents to make crucial decisions on behalf of the broker. In the last chapter, we designed a tariff strategy for our agent that was inspired by the game theory literature. It decided the optimal market share for various player configurations and used heuristic-based techniques to achieve and maintain that market share during the simulation. In this chapter, we replace our heuristics-based strategy with a learning-based strategy to achieve similar performance. For that, we design a tariff strategy that *learns* to achieve and maintain the optimal market share. We model this problem by utilizing techniques derived from *Contextual Multi-armed Bandits* (ConMAB) and solve using the *Exponential-weight algorithm for Exploration and Exploitation (EXP-3)*. Our novelty lies in the formulation of the learning framework; as opposed to previous strategies that aim to maximize profit, we aim to maintain the optimal market share via a learning-based strategy, which in turn reduces other costs and makes our agent profitable. We use ConMAB as its problem setting resembles the tariff generation problem in hand, where, given a context, an agent has to pick an appropriate tariff (an optimal arm of ConMAB) that enables it to maintain the appropriate market share and, in turn, delivers higher returns.

10.2 Tariff Strategy: A Contextual MAB Approach

The retail module of our broker consists of two modules, Tariff Enhancer (TE) and Tariff Designer (TD). The TE submodule comprises the proposed ConMAB-based tariff contract generation strategy, which is solved using the EXP-3 algorithm. This submodule observes the optimal market share for the ongoing game's player configuration by contacting the game theory module explained in Section 9.2, then based on the ConMAB-based learning, it picks the suitable action to enhance the current tariff. This TE sub-module calculates mean tariff rates that would maintain the appropriate market share. The TD sub-module designs weekly ToU tariffs by taking the mean rates suggested by TE as input. The details of the TD sub-module are presented in Section 9.3.2. Below we present the details of the TE module that contains the proposed **GenTariffs-EXP3**.

10.2.1 MDP for the GenTariffs-EXP3

In Section 9.2, we showcase how we determine the optimal market for various player configurations. We also stated that maintaining a market share close to the optimal market share is sufficient to achieve effective profits in the market. Motivated by this, in this section, we showcase the formulation of the proposed GENTARIFFS-EXP3. The proposed strategy is modelled as a Markov Decision Process (MDP) consisting of a tuple $\langle S, A, P, R \rangle$. S represents the state space of the MDP, A denotes the action space and R denotes the rewards of the MDP. P represents the transition probabilities of the MDP, that is, the probability with which MDP transitions to the next state by taking action in the current state. However, the model does not know the transition probabilities. To learn the optimal action in each state (called a policy) in the absence of transition probabilities, we use ConMAB techniques along with the EXP-3 algorithm. Below we describe how the MDP is formulated and optimal policies are learned.

10.2.1.1 The State Space

Here, we define the state space of the GENTARIFFS-EXP3. We construct state space depending on the difference between the current market share (*CMS*) of the GENTARIFFS-EXP3 and the optimal market share (*OMS*) suggested by the game theory module in Section 9.2. Let us denote the difference between both the market shares by Δ , so

$$\Delta = (OMS - CMS)$$

. We categorize Δ into seven buckets, as shown below.

- **State 0:** $|\Delta| \leq OMS * 0.1$
- **State 1:** $\Delta > OMS * 0.1$ and $\Delta \leq OMS * 0.4$
- **State 2:** $\Delta > OMS * 0.4$ and $\Delta \leq OMS * 0.7$
- **State 3:** $\Delta > OMS * 0.7$
- **State 4:** $-\Delta > OMS * 0.1$ and $-\Delta \leq OMS * 0.4$
- **State 5:** $-\Delta > OMS * 0.4$ and $-\Delta \leq OMS * 0.7$
- **State 6:** $-\Delta > OMS * 0.7$

The above state space is designed in such a way that it gives the reflection of the GENTARIFFS-EXP3's current situation in the tariff market. For example, suppose the *OMS* for a game configuration is 50%, then the State 0 occurs when the broker's *CMS* is within $\pm 5\%$ difference of the *OMS* (i.e., between 45% to 55%). Similarly, State 1 happens when the broker's *CMS* is lower than the *OMS*, and the difference between *OMS* and *CMS* ($OMS - CMS$) is more than 5%, but less than 20% (between 30% to 45%). The states 1, 2 and 3 represent the situation when the broker's *CMS* is lower than the *OMS*. Replicating similar logic for the other side, 4, 5, and 6 represent the situation when the broker's *CMS*

is higher than the OMS . The State 4 results in when the difference between CMS and OMS ($-OMS + CMS$) is more than 5%, but less than 20% (between 55% to 70%). The above seven states cover all possible differences between the broker's CMS and the OMS .

10.2.1.2 The Action Space

The action space of the GENTARIFFS-EXP3 generates a new tariff contract in the tariff market. As discussed in Section 4.2.3, a broker needs to come up with rate values to design a new tariff contract. GENTARIFFS-EXP3's action space modifies the currently active tariff or suggests keeping the same tariff active. Below is the action space,

- **Action 0:** $step = 0.0$ [Maintain]
- **Action 1:** $step = -0.02$ [Lower1]
- **Action 2:** $step = -0.04$ [Lower2]
- **Action 3:** $step = 0.02$ [Higher1]
- **Action 4:** $step = 0.04$ [Higher2]

As shown in the action space, GENTARIFFS-EXP3 can choose one of the five actions at any instance. The action selection problem is modelled as a MAB problem, which is solved using **EXP-3** algorithm in Section 10.2.1.4. At any instance, GENTARIFFS-EXP3 can choose to maintain or modify the current tariff. If it chooses to modify the current tariff, it can either decrease the rate value of the currently active tariff or increase the rate value. The above action space provides two options for both scenarios; *Lower1* or *Lower2* to decrease the rate value and *Higher1* or *Higher2* to increase the rate value. After selecting an action, we decide the rate value of the new tariff by adding the *step* value of the selected action to the currently active tariff's average rate value. Thus, a generated new rate value is given to the TD sub-module that designs and publishes the new ToU tariff in the market. Note that, in PowerTAC sign convention, consumption

tariffs are negatively valued as customers need to *pay* that amount; thus, actions such as *Lower1* and *Lower2* would make tariffs more negative (less attractive for customers), and actions such as *Higher1* and *Higher2* would make tariff less negative (more attractive for customers).

10.2.1.3 The Reward Function

The reward function is defined in line with the state space, as shown below.

$$\text{reward} = \begin{cases} 1.00 & , \text{ if } |\Delta| \leq 5\% \\ 0.50 & , \text{ if } |\Delta| \leq 20\% \\ 0.25 & , \text{ if } |\Delta| \leq 35\% \\ 0.00 & , \text{ otherwise} \end{cases}$$

The above reward function awards the GENTARIFFS-EXP3 based on its ability to achieve market share close to the *OMS*. It gets the highest reward of 1 when the absolute difference between the broker's *CMS* and the *OMS* is less than 5%. Similarly, it gets a slightly worse reward when the difference is more than 5% (but less than 20%). The worst case happens when the market share achieved by GENTARIFFS-EXP3 is far away from the *OMS* (the difference is more than 35%); in that case, GENTARIFFS-EXP3 receives a zero reward.

10.2.1.4 The EXP-3 Algorithm

The above contextual MAB-based tariff generation problem is solved using the EXP-3 algorithm. Generally, EXP-3 is used for non-contextual MAB problems but can also be extended for contextual MAB problems. For each state in the state space, It maintains a list of weights for each action in the action space. Using these weights, it stochastically decides which action to take next, and based on the reward received, it increases or decreases the relevant weights. Thus, the generated table resembles the Q-Table in Reinforcement

Learning (RL). In RL Q-Table, the values of the state-action pairs denote how good it is to take that action in the given state in the long run, whereas, in ConMAB, the state-action pairs have the same interpretation, albeit for an immediate future. Due to the similarity, we call the table generated by ConMAB as Q-Table. We introduce an egalitarianism factor $\gamma \in [0, 1]$, tuning the desire to pick an action randomly. That is, if $\gamma = 1$, the weights do not affect the choices at any step. Algorithm 22 shows the modified EXP-3 algorithm for contextual MAB:

Contextual EXP-3

Algorithm 22: Contextual_EXP3(s, t)

Input: State s at time step t

Output: None (just update table of action preferences for state s)

- 1: Initialize/Load table[$|S|$][$|A|$]
- 2: $prob(s, i, t) = (1 - \gamma) \frac{table(s, i, t)}{\sum_{a=1}^{|A|} table(s, a, t)} + \frac{\gamma}{|A|}, \forall i \in \{1, 2, \dots, |A|\}$
- 3: Sample next action act stochastically from
 $[prob(s, 1, t), prob(s, 2, t), \dots, prob(s, |A|, t)]$
- 4: Observe reward $r(s, act, t)$ for taking action act in state s at t
- 5: Update the reward:

$$\hat{r}(s, a, t) = \begin{cases} r(s, a, t) / prob(s, a, t), & \text{if } a = act_t \\ 0, & \text{otherwise} \end{cases}$$

- 6: $table(s, i, t + 1) = table(s, i, t) * e^{\gamma * \hat{r}(s, i, t) / |A|}, \forall i \in \{1, 2, \dots, |A|\}$
-

Algorithm 22 takes the current state s as the input. If the $table$ is empty (at the start of the training), then initialize it with suitable values; otherwise, load the previously created $table$ into memory. As described earlier, the dimensions of this table are $|S| * |A|$ (the size of state space S * the size of action space A). In the next step, we weigh the

actions based on the corresponding values stored in the *table*. The probability of selecting an action i in state s at time t ($\text{prob}(s, i, t)$) is directly proportional to the corresponding state-action pair at time ($\text{table}(s, i, t)$). Here, an egalitarianism factor $\gamma \in [0, 1]$ also plays a role in action selection; $\gamma = 0$ would calculate probabilities purely based on *table* values, while $\gamma = 1$ would assign the same probability to each of the actions. After calculating the probabilities for each action i in state s , in step 3, the algorithm stochastically picks one action based on the calculated probabilities. In step 4, the algorithm observes the reward $r(s, a, t)$ for taking action a in state s at time t . After that, in step 5, the algorithm updates the reward based on whether the action was selected or not; the new reward function $\hat{r}(s, a, t)$ is inversely proportional to the probability $\text{prob}(s, a, t)$. If the action was not selected, then the $\hat{r}(s, a, t)$ is set to zero, as expected. Finally, in step 6, the algorithm updates the *table*; only the state-action pair that got selected at time t gets updated, while other values in *table* remain unchanged. These updates are exponential in nature and proportional to the new reward $\hat{r}(s, a, t)$.

The EXP-3 algorithm deals with the explore-exploit dilemma by stochastically selecting an action based on the calculated probabilities in step 3. This step ensures that the best-known action is picked with higher probability while also occasionally selecting 'not so good' actions. After selecting any action and getting the corresponding reward in that state, the reward is weighed with respect to the probability. A reward for low-probability actions gets enhanced even further, allowing the algorithm to revisit those actions. Thus, the EXP-3 algorithm visits all the state-action pairs a sufficient number of times. In the next section, we show how GENTARIFFS-EXP3 learns the policies to maintain the optimal market shares in each player configuration.

10.3 Deploying GenTariffs-EXP3 in PowerTAC

In this section, we describe how the strategy described in Section 10.2 is deployed to the PowerTAC games. We further demonstrate how the learning process for EXP-3 has

been carried out in the PowerTAC environment. We start by detailing the experimental setup, followed by the results and discussions.

10.3.1 Experimental Set-up

Q-Table Training: As the broker needs to adapt to various player configurations in PowerTAC, we deploy separate tariff MDP and EXP-3 algorithms in each configuration. In this experiment, we train three different models for three-player configurations, namely, 2-Player, 3-Player, and 5-Player. We chose these three configurations as the last PowerTAC tournament (PowerTAC22) had the same configurations. In each player configuration, we played 50 PowerTAC games, where each game simulates the smart grid operations for two months. At the start of the training, we initialize the Q-Table with appropriate values and publish an initial tariff in the market. We keep the same tariff active for a day (24 hours) and then update the tariff at the start of the next day. While updating the tariff, we note the *CMS* and decide the reward to update the Q-Table as shown in Algorithm 22. This constitutes one epoch of training. After that, based on the *CMS*, we calculate the current state and choose an action following the EXP-3 algorithm, and publish a new ToU tariff in the market by using the TD sub-module. We continue this process and record the Q-Table after every checkpoint (typically after every 100 epoch) as well as at the end of the game. While starting a new game, we read and update the previously stored Q-Table while training. We train GENTARIFFS-EXP3 for around 3000 epochs for each configuration and store the final Q-Tables.

Performance Testing: We conduct performance testing to verify whether GENTARIFFS-EXP3 is able to maintain the desired market share during the games after getting trained. As mentioned previously, we store intermediate Q-Tables after every checkpoint and test the effectiveness of GENTARIFFS-EXP3 at various stages of the training. For this, we take Q-Tables from seven different checkpoints, play 10 games with each Q-Table, and record the average market shares during the games. At the end of 10 games, we record the

average and standard deviation of market shares after 10 games; we do this for all seven Q-Tables. In this paper, we present the performance testing for the 3-Player configuration. The following section showcases the results of this experiment.

10.3.2 Results and Discussion

In this section, we present the results of the Q-Table training for the above-mentioned 2-Player, 3-Player, and 5-Player configurations. Furthermore, we also show the efficacy of the GENTARIFFS-EXP3 in maintaining the suggested market share during the games.

Q-Table Training: Figure 10.1, 10.2, and 10.3 are the final Q-Tables after training GENTARIFFS-EXP3 for 50 games (around 3000 epochs) for each player configuration. In Q-Tables, the higher the value (green-shaded region) for any state-action pair, the higher the probability of that action getting selected in the given state.

First, focus on the 2-Player Q-Table in Figure 10.1. GENTARIFFS-EXP3 learns to maintain the currently active tariff if the current state is State 0, which is the best thing to do as the market share is already in the desired range. In State 1 as well, it chooses to continue with the current tariff. When the *CMS* is lower than *OMS* in State 2 and 3, it learns to select *Higher2* action to make tariff cheaper and very much attractive for customers to increase the *CMS* and go closer to *OMS* (*Higher2* would add a high positive step value in the negatively valued tariff, which makes tariff cheaper from customers' perspective). The same explanation is valid for the other side of the state space when the *CMS* is higher than *OMS*. It chooses to *Maintain* in State 4 and go for *Lower2* for remaining states State 5 and 6 in order to make tariff less attractive for customers and decrease the *CMS* and reach closer to *OMS*.

The other two player configurations too converge to similar Q-Tables; however, the values are very different from each other. For example, in State 2, 3-Player Q-Table would select the *Higher2* with high probability, while 5-Player Q-Table would pick *Higher1* or *Higher2* with almost equal probability. In summary, GENTARIFFS-EXP3 learns to decide

Table 10.1: Q-Table for 2-Player Configuration [After 50 Games]

Action State	Maintain	Lower1	Lower2	Higher1	Higher2
0	33.64	16.99	10.35	28.90	14.85
1	361.41	30.35	11.11	18.30	167.86
2	18.07	4.04	3.59	22.95	32.24
3	2.02	1.74	1.32	3.66	7.94
4	40.02	27.96	19.82	20.95	10.69
5	4.33	7.35	15.81	4.31	2.36
6	1.17	2.46	4.47	1.13	1.13

the suitable action in each state for all three player configurations, which empirically looks like the correct action to pick given the state. To prove that the above Q-Tables learn the best actions in each state to achieve the goal of maintaining the desired market, we carried out performance testing for the 3-Player configuration and reported the results below.

Performance Testing: Figure 10.1 shows the market share maintained by Q-Tables stored at various checkpoints (at the 0th epoch, 500th epoch etc.) for a 3-Player configuration. The light blue colour strip in the graph shows the desired market share range for the 3-Player configuration. As seen from the graph, for the 0th epoch Q-Table which has an equal probability for each section getting selected, the market share maintained by GENTARIFFS-EXP3 is very far from the desired range. After 500 and 1000 epochs, too, it is not able to maintain market share in the desired range. However, after getting trained for 1500 epochs, it reaches closer to the desired range. After that, for the higher number of epochs, it maintains the market share within the desired range. The variance (shown

Table 10.2: Q-Table for 3-Player Configuration [After 50 Games]

Action	Maintain	Lower1	Lower2	Higher1	Higher2
State					
0	31.88	31.01	12.77	31.08	16.02
1	56.31	31.82	9.31	46.91	60.29
2	11.92	5.25	3.81	10.75	35.35
3	4.11	1.20	1.30	4.82	24.37
4	26.51	46.61	49.11	23.80	11.94
5	2.91	6.58	4.56	2.74	1.43
6	1.45	2.68	8.40	1.96	1.32

as the bars around the dot) is also low after 2500 epochs of training. A similar result is achieved for the 2-Player and 5-Player configurations as well. This shows the efficacy of GENTARIFFS-EXP3 that learns to update tariffs online and maintains the desired market share during the games.

Table 10.3: Q-Table for 5-Player Configuration [After 50 Games]

Action State	Maintain	Lower1	Lower2	Higher1	Higher2
0	2.30	1.54	1.41	2.00	1.36
1	4.83	2.00	1.47	4.23	2.19
2	2.90	1.45	1.07	6.24	6.40
3	1.72	1.00	1.04	5.85	33.89
4	2.14	5.61	1.78	1.62	1.51
5	1.66	3.60	1.81	1.22	1.10
6	1.36	12.27	10.62	1.44	1.69

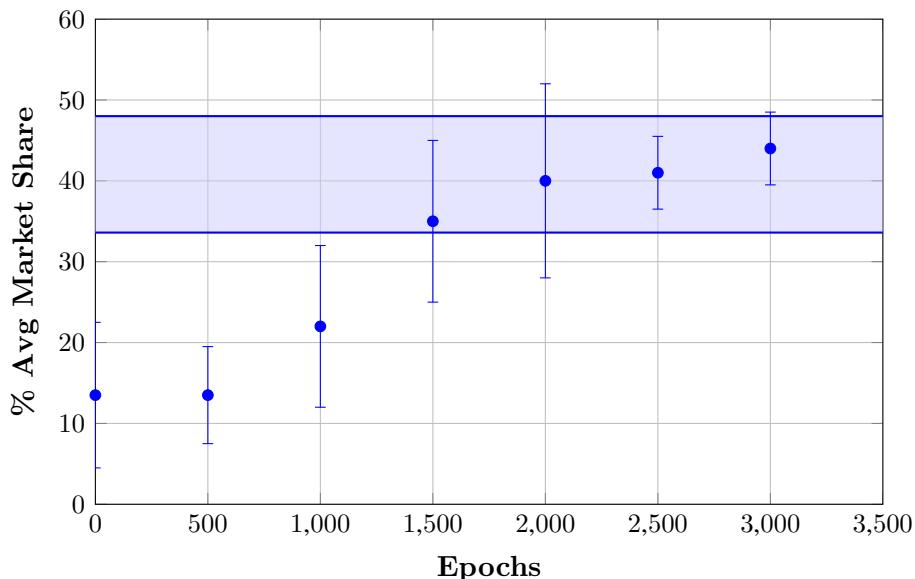


Figure 10.1: Market Share Maintained by GENTARIFFS-EXP3 w.r.t Number of Epochs of Training for 3-Player Configuration

10.4 Summary

Using the Contextual Multi-armed Bandit-based technique, we described the design of an adaptive tariff contract generation strategy, GENTARIFFS-EXP3, to sell electricity in the retail market. In particular, we demonstrated how tariff contracts could be adapted in real-time based on the market situation using the EXP-3 algorithm that efficiently managed the explore-exploit dilemma and visited all the states a sufficient number of times. In our strategy, we first determined the optimal market share and trained GENTARIFFS-EXP3 to achieve and maintain that market share during the game. We showcased that with sufficient training, GENTARIFFS-EXP3 learns the optimal action for a given state and learns to maintain the appropriate market share during the PowerTAC games.

Part B: The Tariff Market

B₂: Designing Tariff-based Demand Response for Peak Reduction in Smart Grid

Chapter 11

Demand Response Model and Method for Peak Reduction

One of the widely used peak reduction methods in smart grids is **demand response**, where one analyzes the shift in customers' (agents') usage patterns in response to the signal from the distribution company. Often, these signals are in the form of incentives offered to agents. This chapter presents the effect of incentives on the probabilities of accepting such offers in a real-world smart grid system with the help of PowerTAC. We first show that there exists a function that depicts the probability of an agent reducing its load as a function of the discounts offered to it. We call it **reduction probability (RP)**. RP function is further parametrized by the rate of reduction (RR), which can differ for each agent. We provide an optimal algorithm, **MJS–ExpResponse**, that outputs the discounts to each agent by maximizing the expected reduction under a budget constraint. When RRs are unknown, we propose a **Multi-Armed Bandit (MAB)** based online algorithm, namely **MJSUCB–ExpResponse**, to learn RRs. Experimentally, we show that it exhibits **sublinear regret**. Finally, we showcase the efficacy of

the proposed algorithm in mitigating demand peaks in a real-world smart grid system using the PowerTAC simulator as a test bed¹.

11.1 Introduction

Load balancing is one of the most prevalent problems in energy grids, which occurs when there is a sudden surge of consumption (i.e., during peak hours) and the demand goes beyond the normal working range of supply. The sudden surge in demand leads to multiple issues: (i) Peak demands put an added load on electricity GenCo to supply additional energy through fast ramping generators to fulfil the energy requirement of the customers (agents). (ii) The grid needs to support such dynamics and peak demand. The ramping up of the generators results in higher costs for the distribution company (DC or broker). Typically, daily peak demands are approximately 1.5 to 2.0 times higher than the average demand [138]. As per one estimation, a 5% lowering of demand during peak hours of California electricity crisis in 2000/2001 would have resulted in 50% price reduction [139]. The same phenomenon is shown in Figure 11.1. Thus, it is paramount to perform load balancing in the grid efficiently.

A promising technology for load balancing is a **smart grid**. It is an electricity network that supplies energy to agents via two-way digital communication. It allows monitoring, analysis, control, and communication between participants to improve efficiency, transparency, and reliability [140]. The smart grid technology is equipped with smart meters capable of handling the load in

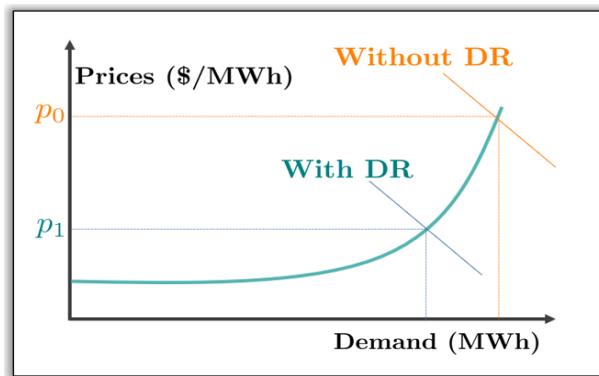


Figure 11.1: Effect of Demand Response

¹This work has been published at AAMAS 2023 and IJCAI 2023.

the smart grid by advising the agents to minimize energy usage during heavy load scenarios. The smart grid system can effectively balance the load by incentivizing agents to shift their energy usage to non-peak timeslots by signalling them the updated tariffs, commonly known as **Demand Response (DR)**.

DR involves brokers offering the agents monetary incentives to optimize their electricity load voluntarily. There are many approaches, such as auction-based mechanisms [118, 119] and dynamic pricing [120] to achieve DR. The major challenge with these approaches is that different agents may respond differently to the given incentives. Thus, to increase agent participation, it becomes crucial to learn their reaction toward these incentives. Learning agents' behaviour is challenging due to the uncertainty and randomness that creep in due to exogenous factors like weather [112, 116]. Works like [112, 116] consider a very simplistic model – when a broker offers an agent incentive more than what it values, the agent reduces every unit of electricity it consumes with a certain probability independent of the incentive. This probability is termed as **Reduction Probability (RP)** [111, 112]. RPs are learned using Multi-armed Bandit (MAB) solutions. There are three primary issues with these approaches. (i) Agents' valuations need to be elicited [111, 112], which adds additional communication complexity, (ii) agents reduce all with RP else nothing, and (iii) RPs do not change with incentives. In the real world, an increase in incentives should lead to an increase in RP. Our work considers the model where the RP is a function of incentives offered and not a constant for an agent, and reduction is not binary.

To model RP as a function of incentive, we need to carry out experiments with smart grids. However, any DR technique (or such experiments) proposed for a smart grid should also maintain the grid's stability. The only way to validate that the proposed technique would not disrupt the grid operations while achieving DR is to test it on real-world smart grids, which is practically impossible. Nevertheless, PowerTAC enables such a scenario and facilitates smart grid research. We first perform experiments with PowerTAC to observe the behaviour of different agents for the offered incentives. With rigorous experiments, we

propose our model **ExpResponse**. We observe that the agents respond quickly to the incentives; however, more incentives may not substantially increase reduction guarantees. Different agents may have a different *rate of reduction (RR)* to incentives that determine how fast RP changes w.r.t. incentives. It also models the consumer valuation for one unit of electricity. A higher RR corresponds to the case where a consumer values the electricity less (for example, a home consumer). In contrast, a lower RR value indicates that the consumer values the electricity higher (for example, an office consumer).

We propose an optimization problem for the broker to maximize the expected peak reduction within the given budget. We then provide an optimal algorithm, namely MJS–EXPRESPONSE, for the case when the RRs of the agents are known. When RRs are unknown, we employ a standard MAB algorithm, MJSUCB–EXPRESPONSE, to learn RRs. Our experiments with synthetic data exhibit sub-linear regret (the difference between the expected reduction with known RRs and the actual reduction with MJSUCB–EXPRESPONSE). With this success, we adopted it for the PowerTAC setup and experimentally showed that it helps reduce peak demands substantially and outperforms baselines, such as distributing the budget equally across all agent segments.

11.2 Preliminaries and Mathematical Model

In a smart grid system, brokers distribute the electricity from GenCo to agents (household customers, office spaces, electric vehicles, etc.) in the tariff market. The customers are equipped with autonomous agents/bots to interact with the grid. Henceforth, we refer to customers as agents. Depending on their type, each agent exhibits a certain usage pattern which is a function of a tariff offered by the broker for most agents. We consider $N = \{1, 2, \dots, n\}$ agents available to prepare for DR at any given timeslot. Here, n denotes the number of types of consumers in the tariff market.

A DR model can further incentivize agents, offering c_i to agent i , to shift their usages from peak to non-peak timeslot. However, agents may do so stochastically, based on

external random events and the offered incentives. For each agent i , this stochasticity can be modelled by associating the probability of reducing demand in the desired timeslot as i . We call this probability as ***reduction probability*** (RP) $p_i(c_i)$. Note that the reduction in electricity consumption at peak slot for agents is ***not binary***. For example, an agent with the usage of 10 KWh and RP ($p_i(c_i)$) of 0.6 would reduce its usage by 6 KWh in expectation. The general intuition is that higher incentives lead to a higher probability of accepting the offer, reducing the load in peak hours. Typically the broker has a limited budget b to offer discounts. It aims to achieve the maximum possible peak reduction within the budget.

First, we need to model the agent's RP function $p_i(\cdot)$. PowerTAC helps us to efficiently model real-world agents' usage patterns and the effects of DR on their usage patterns. Below, we explain the experimental details and observations from the PowerTAC experiments that helped us come up with our novel model of the RP function.

11.2.1 Modelling the Reduction Probability (RP) Function

As presented in Chapter 4, the PowerTAC simulates the smart-grid environments that include different types of customers. However, we focus on PowerTAC's consumption agents, which consume electricity, and aim to learn their RP function.

Experimental Set-up. We perform the following nine sets of experiments to model the RP function. We play 10 different games for 180 simulation days for each experimental set-up and report the statistics averaged over these 10 games. For each experiment, we make the broker publish a tariff at the start of the game and keep the same tariff active throughout the game. The initial tariff rates depend on the broker's electricity purchase cost and may vary from game to game.

FPT Set-up to Identify Peak Slots. We make the broker publish an FPT and record each consumption agent's true usage pattern without any external signals from the broker.

Based on the true usage pattern of each agent, we identify the potential peak demand hours in a day. Figure 11.2 shows the usage pattern of a PowerTAC agent in response to the FPT; in this figure, the hours 7 and 17 have the peak usages during the day. The rate value of the FPT is derived by adding a profit margin to the broker's electricity purchase cost. Next, we study the agents' responses to different tariffs. To this, we consider the ToU model, where different prices are proposed at different times. These prices, however, are the same for all agents.

ToU Set-up. In ToU tariffs, the rate charged for each unit of electricity consumed can vary depending on the time of the day. The ToU tariffs are designed so that the agents get discounts during non-peak hours and no/little discounts during peak hours. The average rate of the tariffs across all timeslots remains the same as the previous FPT setup. Essentially, all the ToU tariffs have the exact same area under the curve (AUC) as the FPT. We perform such an experiment for the remaining 8 sets by offering discounts in each set; we give $x\%$ discount on non-peak timeslots compared to the price in peak timeslots. Here $x \in \{1, 2, 5, 7.5, 10, 15, 20, 30\}$. Figure 11.2 explains how we move from an FPT (Fig. 11.2(a)) to a ToU tariff (Fig. 11.2(c)) by offering a certain discount and keeping the AUC the same for all the tariffs. Based on the discount level, the agents modify their usage patterns ((Fig. 11.2(b,d))), and we collect the usage data of each agent for each of the sets.

To analyze the effects of various discounts on agents' usage patterns, we pick the top two peak hours in the day for each agent. Then, we calculate the difference between the electricity usage during FPT-Set-up and electricity usage during discounted ToU-Set-up for both peak slots. We do this for all eight sets numbered from 2 to 9. We can view the discounted tariffs as a DR signal for the agents to shift their non-priority usages from peak to non-peak timeslots. Below, we show the observations of the DR experiments for a few selected agents.

Figure 11.3(a) and Figure 11.3(b) show the DR behaviour of three PowerTAC agents BrooksideHomes, CentervilleHomes, and EastsideOffices for their top two peak demand

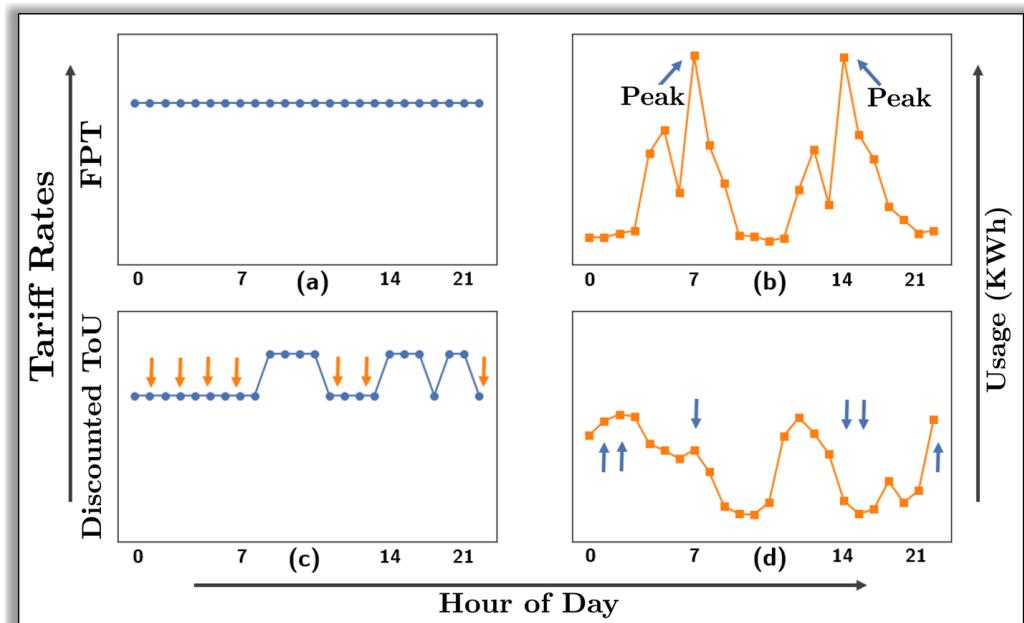


Figure 11.2: PowerTAC customer's response for FPT and ToU tariffs

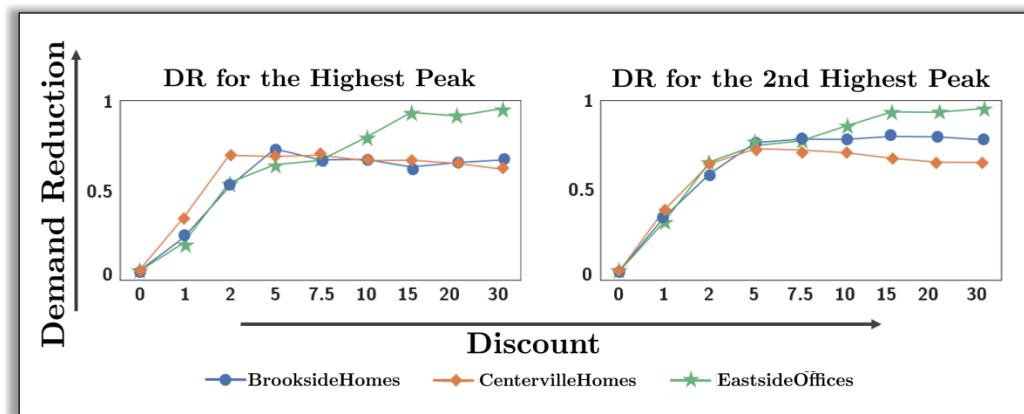


Figure 11.3: DR Probability Function of PowerTAC Customers for (a) The Highest Peak (left), (b) The 2nd Highest Peak (right)

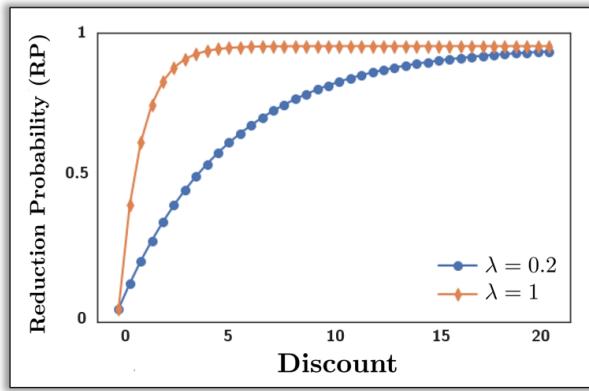


Figure 11.4: Exponential Probability Function for EXPRESPONSE

hours (re-scaled to visualize peak reduction as a probability function). The first two agents are household customers, whereas the last agent is an office customer. Analysing the plots gives a crucial insight into the agents' behaviour. The agents reduce their usage by a great extent for the initial values of discount (1%, 2% and 5%) but cannot reduce their usage further even when offered a much higher discount; Secondly, different agents follow the different rate of reduction.

Based on the PowerTAC experiments, we conclude that the reduction probability function can be modelled by an exponential probability function and is given as:

$$p_i(c_i) = 1 - e^{-\lambda_i c_i}, \forall i \in N \quad (11.1)$$

Here, c_i is a discount (or incentive) given to agent i , and λ_i is its reduction rate (RR). The proposed function depends upon the choice of λ_i ; the higher value of λ_i generates a steeply increasing curve (as shown with $\lambda_i = 0.5$), while the lower λ_i value makes the curve increase slowly with each discount (as shown with $\lambda_i = 0.1$) as shown in Figure 11.4. Let $c = (c_1, c_2, \dots, c_n)$ and $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_n)$ be vector of offered incentives and RRs.

11.2.2 ExpResponse: The Optimization Problem

We assume that all the agents have the same electricity consumption in peak slots². The aim is to maximize the expected reduction under a budget constraint. This leads to the following optimization problem:

$$\max_{c_i} \sum_{i=1}^n (1 - e^{-\lambda_i c_i}) \text{ s.t. } \sum_{i=1}^n c_i \leq b \quad (11.2)$$

Thus, the right-hand side of the equation is the final objective function that needs to be maximized subject to budget constraints on the amount of available discounts with the broker. Suppose the RR (λ) values are known. In that case, we present an optimal algorithm MJS–EXPRESPONSE to efficiently distribute the budget b among the agents to maximize the expected sum of peak reduction (Section 11.3.1). When RRs are unknown, we provide MJSUCB–EXPRESPONSE algorithm that estimates it (Section 11.3.2). The algorithm is motivated by multi-armed bandit literature [111, 112] and uses the linear search over the possible range of values of RR.

11.3 Proposed Algorithms for ExpResponse

This section proposes a novel algorithm to solve EXPRESPONSE. We discuss two settings: (i) perfect information that assumes the knowledge of RR and (ii) imperfect information where RR values need to be learned over time.

11.3.1 Perfect Information Setting: Known RR

MJS–EXPRESPONSE (Algorithm 23) distributes one unit of budget to an appropriate agent in each iteration until the entire budget is exhausted. To decide the appropriate

²Agents consuming different amounts can be trivially modelled by duplicating agents

agent for the current iteration, we calculate *jump* (Δ) values for all the agents. We define Δ_i^{j+1} value for each agent as the change in RP for a unit change in discount. For example, if an agent i has RP of $(1 - e^{-\lambda_i(j+1)})$ for discount $c_i = (j + 1)$ and RP of $(1 - e^{-\lambda_i j})$ for discount $c_i = j$, then the jump Δ_i^{j+1} is the difference between these two probabilities.

MJS-ExpResponse

Algorithm 23: MJS_ExpResponse(b, n, λ)

Input: Budget b , number of agents n , reduction rate (RR) vector λ

Output: Final Allocation vector c

```

1:  $cost \leftarrow 0, c \leftarrow 0_n$ ; # initialization
2: while  $cost \leq b$  do
3:    $d \leftarrow 1$  # iterator
4:    $l \leftarrow 1$  # index of agent with largest jump
5:   while  $d \leq n$  do
6:      $\Delta_d^{c_d+1} \leftarrow (1 - e^{-\lambda_d(c_d+1)}) - (1 - e^{-\lambda_d c_d})$ 
7:      $\Delta_l^{c_l+1} \leftarrow (1 - e^{-\lambda_l(c_l+1)}) - (1 - e^{-\lambda_l c_l})$ 
8:     if  $\Delta_d^{c_d+1} > \Delta_l^{c_l+1}$  then
9:        $l = d$ 
10:      end if
11:       $d = d + 1$ 
12:    end while
13:     $c_l = c_l + 1$  and  $cost = cost + 1$ 
14:  end while
15: return  $c$  # final allocation

```

The algorithm finds an agent l having the maximum such jump for the current unit of reduction (denoted by $\Delta_l^{c_l+1}$ for agent l) and allocates the current unit discount to agent

l–Maximum Jump Selection (MJS). The Problem can be rephrased as finding these jumps that can fetch the highest increase in probabilities. At the end of the inner while loop, we allocate the unit discount to the agent having the highest jump for the current unit of reduction and increase the total allocated budget till now. Finally, the algorithm returns the allocation, which is the optimal distribution of the initial fixed budget, as shown in the below theorem.

Theorem 11.1. MJS–EXPRESPONSE *is optimal.*

Proof. For any discount vector c , the objective function in Equation 11.2 can be written as a sum of jumps Δ_i^j which denote the additional increase in reduction probability of consumer i when offered j units of discount compared to $j - 1$. i.e.

$$\begin{aligned} & \max_c \sum_{i=1}^n 1 - e^{-\lambda_i c_i} \\ &= \max_c \sum_{i=1}^n \left(\sum_{j=1}^{c_i} (1 - e^{-\lambda_i j}) - (1 - e^{-\lambda_i (j-1)}) \right) \\ &= \sum_{j=1}^{c_i} \Delta_i^j \max_c \sum_{i=1}^n \sum_{j=1}^{c_i} \Delta_i^j \quad s.t. \quad \sum_{i=1}^n c_i \leq b \end{aligned}$$

Thus, at the optimal solution, one unit is allocated to b highest jumps and 0 to other jumps. We now need to prove that the earlier jump is higher than the latter, i.e., $\Delta_i^l \geq \Delta_i^j \forall l < j$. Lemma 11.1 proves this for any agent i . \square

Note that one can use KKT conditions and derive a set of linear equations to determine an optimal distribution of b . Our proposed algorithm is simple, determines an optimal solution in linear time, and has a time complexity of $O(nb)$. However, in most cases, agents' RRs are not known initially, and it needs to be learned by optimally distributing the budget among competing agents. The next section describes our novel algorithm MJSUCB–EXPRESPONSE to estimate RRs.

Lemma 11.1. For each i , we have $\Delta_i^j \geq \Delta_i^{j+1}$ with $\lambda_i \geq 0$

Proof. We have the following:

$$\begin{aligned} e^{-j\lambda_i}(e^{\lambda_i} - 1) &\geq e^{-j\lambda_i}(1 - e^{-\lambda_i}) \\ \Rightarrow e^{-\lambda_i(j-1)} - e^{-\lambda_ij} &\geq e^{-j\lambda_i} - e^{-\lambda_i(j+1)} \end{aligned}$$

From the last equation, we have $\Delta_i^j \geq \Delta_i^{j+1}$. \square

11.3.2 Imperfect Information Setting: Unknown RR

As the RRs of the agents are unknown in this setting, we estimate them based on the history of the agents, which consists of the agents' responses during the past timeslots. For each agent, we store its historical behaviour by keeping track of the offered history and success history; we estimate $\hat{\lambda}$ and $\hat{\lambda}^+$ through a routine *LinearSearch*(\cdot).

MJSUCB–ExpResponse. We start by initializing $\hat{\lambda}$ and its UCB component $\hat{\lambda}^+$, then estimating $\hat{p}_i(c_i)$ for each agent i and for each c_i using *Linearsearch*(\cdot) at each timeslot.

LinearSearch(). Estimating RRs with the offered history and success history, we calculate $\hat{p}_i(c_i) = \frac{\text{SuccHist}(i, c_i)}{\text{OfferredHist}(i, c_i)}$, for each c_i offered to the agent i . $\hat{p}_i(c_i)$ is then used to calculate candidate values of RR using Equation 11.1. Based on the candidate RR values, we determine $\hat{\lambda}_i$ that minimizes the squared error loss between the historical probabilities and the probabilities calculated based on the Equation 11.1 i.e.,

$$\hat{\lambda}_i \in \operatorname{argmin}_l \sum_{c_i \in [b]} \left(\hat{p}_i(c_i) - (1 - e^{-lc_i}) \right)^2, \forall c_i$$

MJSUCB–ExpResponse

Algorithm 24: MJSUCB_ExpResponse(b, n, bS, T)

Input: Budget b , number of agents n , batch Size bS , number of iterations T

Output: Allocation $\{c_t\}_{t=1}^T$

```

1: Initialize  $\hat{\lambda}, \hat{\lambda}^+$  randomly #  $n$ -dimensional vectors
2: Initialize  $offeredInst, successInst, offeredHist$  and  $successHist$  to 0
    # 2D matrices of size  $n \times b$ 
3:  $t \leftarrow 0$ 
4: while  $t < T$  do
5:    $\{c_{t'}\}_{t'=t}^{t+bS} = \text{MJS-EXPRESPONSE}(b, n, \hat{\lambda}^+)$ 
6:   for  $i = 1 \rightarrow n$  do
7:     if  $c_t(i) \neq 0$  then
8:        $offeredInst(i, c_t(i)) += bS$ 
9:        $successInst(i, c_t(i)) +=$  # Successes for agent  $i$ 
10:      end if
11:    end for
12:    Update  $Hist = \{Hist, offeredInst, successInst\}$ 
13:    Clear  $offeredInst, successInst$ 
14:     $t \leftarrow t + bS$ 
15:     $[\hat{\lambda}, \hat{\lambda}^+] \leftarrow LinearSearch(Hist, n, b, t)$ 
16:  end while
17: return  $\{c_t\}_{t=1}^T$ 

```

The RR value that achieves the least squared error loss is returned as the optimal RR after the current timeslot. We follow the same method for each of the agents. Algorithm 24 discusses our proposed MJSUCB–EXPRESPONSE method in more detail, which takes bud-

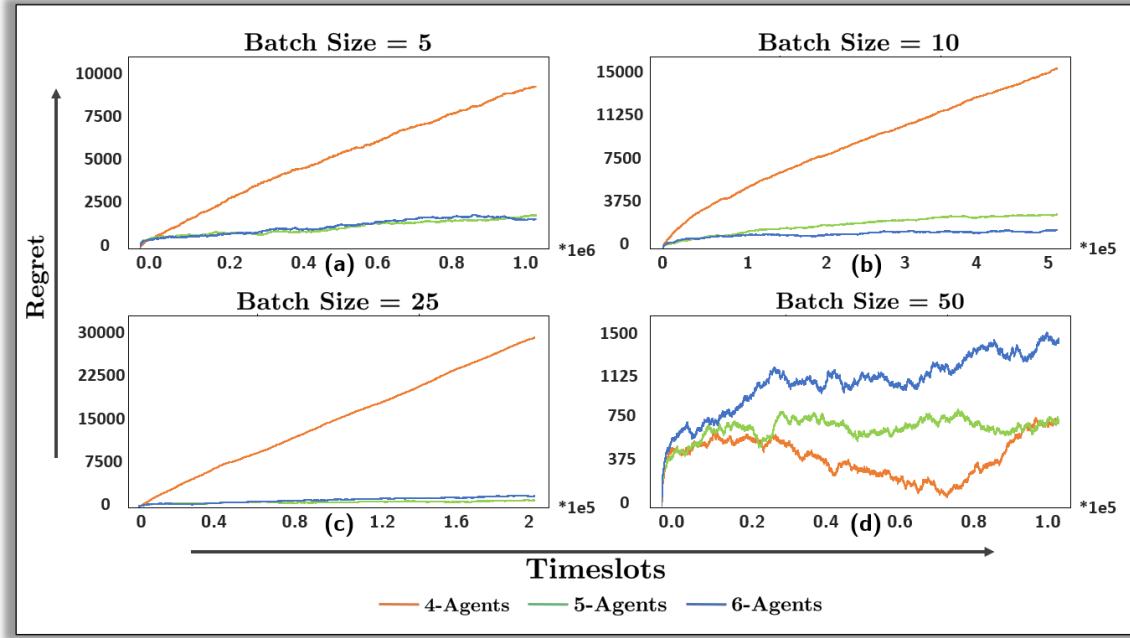


Figure 11.5: Comparing MJSUCB–EXPRESPONSE’s regret over 25 iterations with varying batch sizes [budget=5, T=5M]

get b , n , batch Size bS , and T as inputs and returns $\hat{\lambda}$ s. Here, $\hat{\lambda}, \hat{\lambda}^+$ denote estimated λ and its UCB version, respectively.

11.3.3 Experimental Evaluation of MJSUCB–ExpResponse

To check whether the algorithm converges to the true RR, we conduct extensive analysis on a simplified version of a smart grid. Here, we discuss the experimental set-up to observe the regret of the proposed MJSUCB–EXPRESPONSE. Regret is the difference between total reduction with known RRs and total reduction with unknown RRs. In both experiments, we repeat the experiment 25 times, each instance having independently chosen λ . We report different statistics averaged over 25 iterations.

Exp1– Effect of Batch Size. In this experiment, we keep the budget b and T constant, and vary batch sizes. This experiment shows the change in regret behaviour as we change the batch sizes from low to high. Figure 11.5 compares the average regret of MJSUCB–EXPRESPONSE over 25 iterations of varying true RR values of agents, with varying batch size, $b = 5$, and $T = 5000000$. The figure shows four subplots with batch sizes of 5, 10, 25, and 50, respectively. Each subplot compares the regret values for 4, 5, and 6 agents, respectively, and shows *sub-linear* regret in the case of 5 and 6 agents for four different batch sizes. With an increased batch size to 50, even the case with 4 agents converges to sub-linear regret within a few timeslots.

Exp2– Effect of Budget and Relation w.r.t. T . In this experiment, we vary the budget and number of agents while keeping the number of iterations and time T constant. Figure 11.6 compares the *regret* and concludes that MJSUCB–EXPRESPONSE achieves a sub-linear regret, and its peak reduction success rates are approximately the same as the optimal peak reduction success rates. We next show the performance of MJSUCB–EXPRESPONSE in PowerTAC.

11.4 Adapting MJSUCB–ExpResponse for PowerTAC

First, we explain how we model different customer groups in PowerTAC.

Modelling the Customer Groups. We begin by grouping the agents based on their electricity usage and create four groups, namely, $G1$, $G2$, $G3$, and $G4$, as shown in Table 11.1. We consider groups due to the limitations of PowerTAC, where we cannot publish individual customer-specific tariffs. We can only publish tariffs for customer groups having similar usage ranges. However, our proposed model and algorithms do not rely on any assumption of the existence of such groups and treat each consumer as a separate user (in

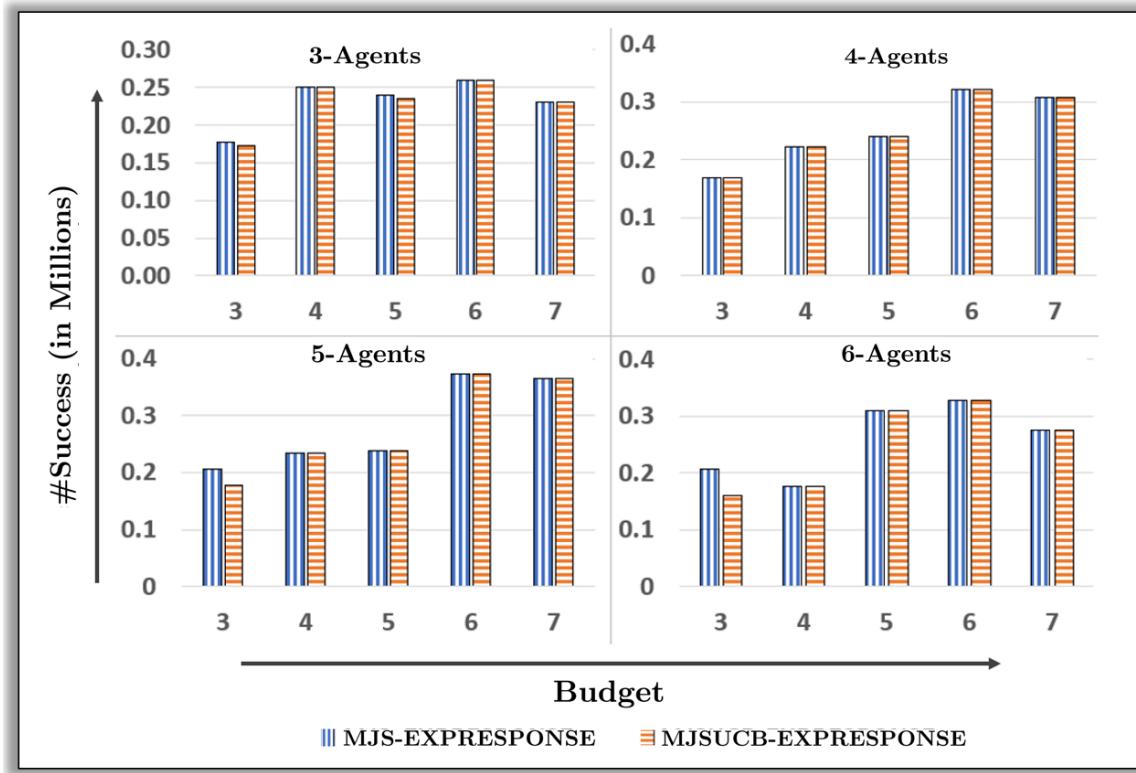


Figure 11.6: Comparing MJS-EXPRESPONSE and MJSUCB-EXPRESPONSE with varying budget and number of agents [Iterations = 25, T=5M]

Sections 11.2 and 11.3). We leave out the remaining PowerTAC agents as they do not use a considerable amount of electricity in the tariff market.

Designing the Tariffs for Each Group. For each group G_i , we publish ToU tariff $ToUi$ such that agents from G_i subscribe to tariff $ToUi$, and no other group of agents subscribes to that tariff. To achieve this, we combine ToU tariffs with **tier** tariffs as follows. In PowerTAC, tier tariffs specify rate values and upper bounds on electricity usage below which the specified rates are applicable. However, if the usage goes beyond that particular bound, the agent has to pay the rate values associated with the next higher bound. As we have segregated the agents based on their usage range, for any targeted group, we offer

Table 11.1: Customer groups detail

Group	Customers	Type	%Usage in Market
$G1$	BrooksideHomes & CentervilleHomes	Household	50%
$G2$	DowntownOffices & EastsideOffices	Small Offices	25%
$G3$	HextraChemical	Mid-level Offices	10% to 12%
$G4$	MedicalCenter-1	High-level Offices	10% to 12%

standard ToU rate values for its particular usage range and high rates for the remaining ranges of electricity usage. Thus, a group of agents naturally like the tariff designed for their group as the other tariffs are way costlier for their usage pattern. At any moment in the PowerTAC game, we keep all four *ToU* tariffs active (one for each group); these tariffs keep getting updated based on the DR signals from the broker.

Adapting MJSUCB–ExpResponse in PowerTAC. While proposing our model, we assume that agents are identical and have the same usage capability. Thus, maximizing the sum of probability would also result in maximizing reduction. However, for general smart grid settings such as PowerTAC, we modify our model by giving weightage to agents based on their usage percentage (market cap). Higher weightage is given to agents that can reduce the larger amount of energy. We modify MJS–EXPRESPONSE to introduce weights proportional to groups' contribution to electricity usage for each group. $weights = \{4, 2, 1, 1\}$ to groups $\{G1, G2, G3, G4\}$, respectively in our experiments. We still use $c = \text{MJS–EXPRESPONSE}(\cdot)$ (Line 5, Algorithm 24) to find the group that can fetch the highest increase in the probabilities as shown in Algorithm 23. While allocating discounts to the groups, instead of allocating a 1 unit of budget to each group, we weigh the unit with the group's weight. For example, if $G1$ gets selected for the discount, we assign a 4 unit discount instead of 1. We call this way of allocation as WEIGHTEDMJS–EXPRESPONSE. It may help to assign weights to the groups as assigning weights will allocate discounts

proportional to their peak reduction capacity. For instance, 10% reduction in $G1$ would reduce more peak demand than 10% reduction in $G4$.

Creating Baseline. To compare the performance, we consider the baseline of uniformly allocating the budget to all the groups. This leads to publishing group-specific tariffs with equal discounts. We record the peak reduction efficiency and reduction in capacity transactions from the baseline strategy. We then use the recorded information as a benchmark to evaluate MJSUCB–EXPRESPONSE performance. Furthermore, we compare the efficacy against the strategy when we do not provide groups with any DR signals.

Evaluation Metrics. Finally, we evaluate MJSUCB–EXPRESPONSE’s performance on two metrics, (i) MJSUCB–EXPRESPONSE’s peak demand reduction capability, which indicates how much percentage of peak demand reduction MJSUCB–EXPRESPONSE achieved compared to the benchmark strategies, and (ii) the reduction in ***capacity transaction*** penalties that suggest how effectively MJSUCB–EXPRESPONSE can mitigate such penalties compare to the benchmark strategies.

Capacity Transactions. In PowerTAC, capacity transactions are the penalties incurred by the broker if the agents subscribed to their portfolio contribute to the peak demand scenarios. These huge penalties are a way to penalize the broker for letting the agents create demand peaks. Thus, as opposed to the previous section where we analytically show MJSUCB–EXPRESPONSE exhibits a sub-linear regret, here in PowerTAC experiments, we aim to reduce capacity transaction penalties of the broker using MJSUCB–EXPRESPONSE.

11.4.1 Experiments and Discussion

Experimental Set-up. We perform multiple experiments with different initial budgets. We play 8 games in each set with approximately 28 simulation weeks (total 210 weeks). For each set, we start the experiments by randomly initializing RR values for each group and

calculate the budget allocation based on WEIGHTEDMJS–EXPRESPONSE as well as MJS–EXPRESPONSE (line 5 in MJSUCB–EXPRESPONSE), called as MJSUCB–EXPRESPONSE-W and MJSUCB–EXPRESPONSE-UW, respectively.

As explained in Section 11.4, for each of the 4 groups, we have four ToU tariffs. We keep the same tariffs active for 3 simulation days and invoke the MJSUCB–EXPRESPONSE at the end of the 3rd day. Based on the success probabilities, we update the *offeredHist* and *successHist*, and calculate the next set of $\hat{\lambda}$ and $\hat{\lambda}^+$ values for each group. Using the $\hat{\lambda}^+$, we calculate the next demand allocation and publish the new tariffs as explained earlier. While publishing new tariffs, we revoke the previous ones; thus, only 4 tariffs are active at any time in the game. This 3 days process constitutes a single learning iteration (t).

To calculate the success probability of each tariff, we played 10 offline games without any discount to any group and noted down the top two peak timeslots. Let $x_{i,1}$ and $x_{i,2}$ denote per group usage during those peak timeslots. Then, we compute the success probability as $p_{i,1} = (1 - y_{i,1}/x_{i,1})$ and $p_{i,2} = (1 - y_{i,2}/x_{i,2})$, with $y_{i,1}$ and $y_{i,2}$ denoting group 1 and 2 usage respectively. p_i is then set as $p_i = \frac{p_{i,1} + p_{i,2}}{2}$. We perform 2 sets of experiments with $b = 15\%$ and $b = 7.5\%$. We define a scalar value that gets multiplied by the discounts to generate fractional discounts.

Observations and Discussion. Table 11.2 shows the cumulative peak usages under MJSUCB–EXPRESPONSE and bench-marked (baseline) method for the top 2 peaks of groups $G1$ to $G4$. As shown in the table, for the overall 210 simulation weeks of training in PowerTAC, MJSUCB–EXPRESPONSE cumulative peak usage for peak1 is similar to the baseline method for both weighted and unweighted allocations while slightly worse than the baseline for peak2. The observation is consistent for the budget values 15% and 7.5%.

However, if we focus on only the last 10 weeks of training, MJSUCB–EXPRESPONSE’s peak usage reduction capabilities are visible. Both weighted and unweighted allocations achieve cumulative peak reduction close to 14.5% concerning **No Discount** peak usages for peak1 and $b = 15\%$, which is almost 5 times better than the baseline while maintain-

Table 11.2: Peak usage comparison (usage in MWh)

Method	$b = 15$		$b = 7.5$	
	P1	P2	P1	P2
No Discount	70.2	69.7	70.2	69.7
Baseline	68.1	67.4	67.8	67.7
Average Over All 210 Weeks of Training				
MJSUCB–ExpResponse–W	68.4	67.8	68.8	69.5
MJSUCB–ExpResponse–UW	68.3	68.1	68.1	68.6
Average Over Last 10 Weeks of Training				
MJSUCB–ExpResponse–W	60.2	69.8	64.1	70.9
MJSUCB–ExpResponse–UW	60.0	67.6	61.4	69.0

ing similar performance as the baseline for peak2. Similarly, MJSUCB–EXPRESPONSE achieves significant improvement for $b = 7.5\%$ too for peak1 by reducing the peaks 3 to 4 times better than baseline.

Furthermore, as shown in Table 11.3, capacity transaction penalties in the last 10 weeks are significantly lower than the **No Discount** and baseline. Due to DR signals, agents sometimes shift some of the demand from peak1 to peak2 or cannot reduce any demand from peak2. However, if the overall system’s performance is observed with the help of capacity transaction penalties in PowerTAC experiments, the penalties are significantly lower than the baseline, reinforcing the efficacy of MJSUCB–EXPRESPONSE in the PowerTAC environment.

Table 11.3: Capacity Transaction Comparison (Average Penalty)

Method	$b = 15$	$b = 7.5$
No Discount	249355	249355
Baseline	233768	227070
Average Over All 210 Weeks of Training		
MJSUCB-ExpResponse-W	233643	228478
MJSUCB-ExpResponse-UW	233248	229467
Average Over Last 10 Weeks of Training		
MJSUCB-ExpResponse-W	226374	228351
MJSUCB-ExpResponse-UW	225775	228276

11.5 Summary

The paper proposed a novel DR model where the user’s behaviour depends on how many incentives are given to the users. Using the experiments on the PowerTAC real-world smart grid simulator, we first showed that agents’ probability of accepting the offer increases exponentially with the incentives given. Further, each group of agents follows a different rate of reduction (RR). Under the known RR setting, we proposed MJS-EXPRESPONSE which leads to an optimal allocation of a given budget to the agents, which maximizes the peak reduction. When RRs are unknown, we proposed MJSUCB-EXPRESPONSE that achieves sublinear regret on the simulated data. We demonstrated that MJSUCB-EXPRESPONSE is able to achieve a significant reduction in peak demands and capacity transactions within just 200 weeks of simulation on the PowerTAC simulator.

Chapter 12

VidyutVanika in PowerTAC Tournament

PowerTAC organizes an annual tournament where multiple teams participate by designing their autonomous brokers to compete with other teams. We participate in this tournament with the broker called **VidyutVanika**. VidyutVanika is equipped with combinations of the strategies described in the previous chapter. In this chapter, we present the details of our broker, VidyutVanika, in the PowerTAC tournaments in the years 2021 and 2022. We describe the architectures of both brokers, along with a detailed analysis of their performances in the PowerTAC tournaments¹.

12.1 Journey of VidyutVanika in the PowerTAC

PowerTAC aims to bring the digital revolution to the agent-based operations of smart grids; thus, it encourages researchers to come up with novel ideas to design autonomous intelligent brokers that can deal with complex interactions in smart grid systems. To achieve this objective, PowerTAC enabled a competition environment in its smart grid

¹A part of this work has been published at IJCAI 2022.

simulator, where multiple teams of researchers participate by deploying their autonomous brokers. The goal of each team is to design a broker who can earn the highest cumulative profits and win the tournament while contributing to smart grids and autonomous agents' literature with novel ideas.

One such broker in PowerTAC tournaments is our broker, VidyutVanika, and our team is called **Team VidyutVanika**. The word “*VidyutVanika*” is a combination of two Sanskrit words, “**Vidyut**” and “**Vanika**”. In Sanskrit, “**Vidyut**” means electricity, and “**Vanika**” means broker, so the name “**VidyutVanika**” translates to “**Electricity Broker**”.

PowerTAC has organized these tournaments since the year 2012. In the following sections, we present the details about the VidyutVanika that participated in PowerTAC tournaments in 2021 and 2022.

12.2 VidyutVanika: Overview

In this section, we present a generic architecture of VidyutVanika (VV), followed by architectures of VidyutVanika21 (VV21) and VidyutVanika22 (VV22). VV21 and VV22 are the brokers deployed in the 2021 and 2022 PowerTAC tournaments, respectively. Figure 12.1 shows the generic architecture of VV, which consists of a wholesale and a tariff module. The wholesale module is responsible for submitting bids and asks in a PDA of the wholesale market. The tariff module is in charge of publishing or revoking tariffs in the tariff market. Additionally, VV maintains dedicated repositories to store data from the PowerTAC game server, including the weather, wholesale, balancing, and tariff market information. VV uses this stored information in the decision-making process. The strategies used by VV in the wholesale and tariff module varies tournament to tournament, which we highlight in the next paragraph.

VV21 relies on AI and GT to design its wholesale and tariff strategies. VV21 follows the supply curve of the prominent GenCo in the wholesale market and designs its wholesale

bidding strategy based on the learned supply curve model. For this, as depicted in Figure 12.2, it uses Demand Predictor (DP), Supply Curve Follower (SCF), and Bid Generator (BG) sub-modules. While in the tariff market, VV21 determines the optimal market share using GT analysis and designs tariffs to maintain such market share with the help of Tariff Enhancer (TE), Tariff Designer (TD), and Tariff Health Checker (THC) sub-modules. These sub-modules are detailed in Section 12.3.

On the other hand, VV22 uses VV21’s wholesale and tariff modules in some games, while in some games, it uses learning-based strategies in both wholesale and tariff markets. More precisely, in those games, VV22 uses a DDPG-based bidding strategy to place bids in the wholesale market, while, in the tariff market, VV22 deploys a multi-armed bandit-based tariff strategy, as shown in Figure 12.8. The other sub-modules, like Demand Predictor (DP) and Bid Generator (BG) in the wholesale module and Tariff Designer (TD) in the tariff modules, remain the same as VV21. These sub-modules are detailed in Section 12.4.

In the next sections, we present the details about the PowerTAC2021 and PowerTAC2022 tournaments, as well as the architectures of our brokers, VidyutVanika21 and VidyutVanika22, that were deployed in these tournaments.

12.3 VidyutVanika21 in the PowerTAC2021 Tournament

The PowerTAC2021 tournament was organized in September–October 2021. The tournament was preceded by a few trial rounds to help teams test and rectify their autonomous broker’s strategies. The qualifier round of PowerTAC2021 happened in September, in which a total of 10 teams participated; the qualifiers lasted for 16 days. At the end of the qualifier round, 7 teams out of 10 qualified for the finals; VidyutVanika21 topped the qualifier round.

The final of PowerTAC2021 took place from 11 to 27 October, 2021, in which 7 qualified teams participated. A total of 386 games played between 7 brokers in 7-Player, 5-Player, and 3-Player configurations. In the 7-Player configuration, 7 brokers are present in a game;

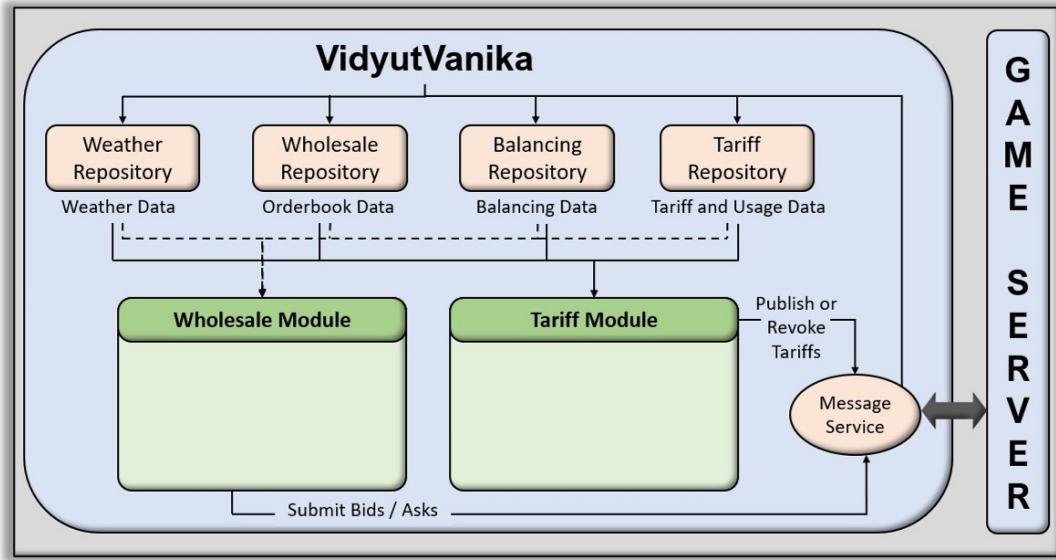


Figure 12.1: An Abstract Architecture of VidyutVanika

similarly, for 5-Player and 3-Player configurations, 5 and 3 brokers compete against each other in a game, respectively. A total of 50 7-Player, 90 5-Player, and 90 3-Player games were organized in the finals of PowerTAC2021; thus, each broker played exactly 230 games in the finals. The games happened in three different weather locations, namely, Denver, Minneapolis, and Phoenix; the weather location is randomly chosen at the start of the game. The weather location affects the customers' electricity usage pattern, clearing prices of the auctions, among various other variations. For instance, during the summertime in the Phoenix region, the electricity usage of customers goes extraordinarily high, which creates a deficit of electricity, and thus electricity prices surge to record high values. So, a broker needs to carefully adjust its strategies depending on various player configurations as well as various weather configurations. Below, we discuss the architectural details of our broker, VidyutVanika21, who participated in the PowerTAC2021 tournament.

12.3.1 Design and Overview of VidyutVanika21

As briefed in the overview, VidyutVanika21 (VV21) deploys game theory and AI-based strategies to operate in the wholesale and tariff markets of PowerTAC. Figure 12.2 shows the architecture of VidyutVanika21. The wholesale module is denoted as VV21-WM, and the tariff module is called VV21-TM. VV21-WM is responsible for energy procurement in the wholesale market, while VV21-TM handles tariff publication and revocation in the tariff market. Additionally, VV21 maintains repositories to store data received from the game server about the weather, wholesale, balancing, and tariff markets, which is then used in the decision-making process.

For effective bidding in the wholesale market, VV21-WM follows the supply curve of GenCo to recognize its lowest ask for a particular auction. It uses this lowest ask as an upper bound on its limit price and tries to procure as much power from other sellers present in the wholesale market at lower prices than what is demanded by GenCo; the strategy is discussed in detail in Chapter 6. The strategy used by VV21-TM in the tariff market is inspired by game theory and aims to maintain an appropriate level of market share in the tariff market. Furthermore, the tariff module also targets achieving a balanced portfolio consisting of customers of different types (consumption, production, and storage). Tariffs usually contain surcharges during peak hours to further mitigate the effect of capacity transaction charges; the tariff module of VV21 is presented in Chapter 9. Finally, our design of VV21-WM and VV21-TM helps in handling balancing market activities as well. Precisely, VV21-WM procures a large share of energy from the wholesale market, thus reducing purchases from the balancing market, which results in low balancing costs. During high-demand scenarios, when GenCo cannot fulfil all of the market demand, VV21-TM supplies energy via producers and storage customers in its portfolio to make up for the market deficit and, in the process, generates revenue. As the details of both modules are already discussed in the above-mentioned chapters, we provide a brief description of all the sub-modules below.

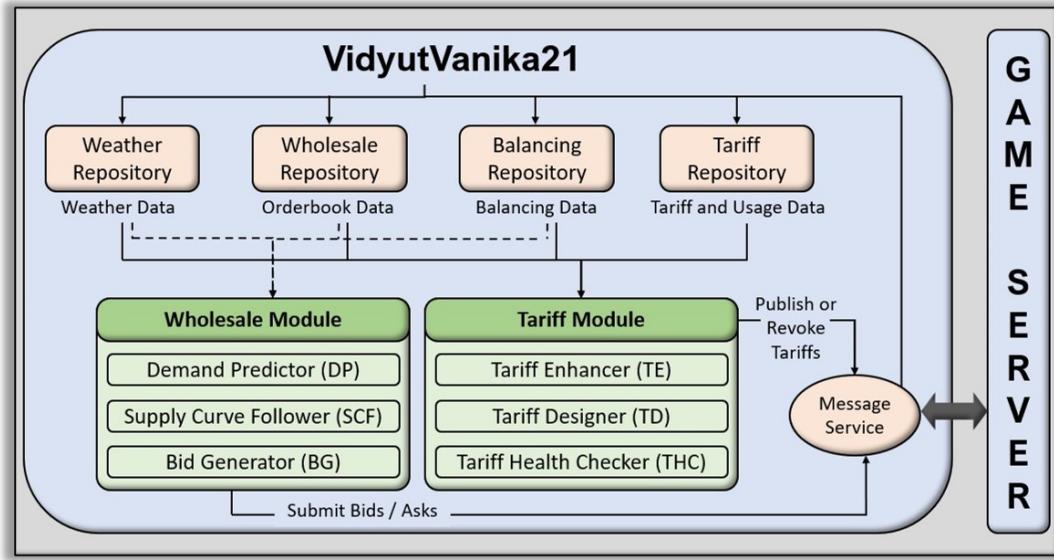


Figure 12.2: VidyutVanika21: System Architecture

12.3.2 Wholesale Module of VidyutVanika21

The wholesale module of VV21 (VV21-WM) aims to minimize energy procurement costs while buying needed electricity for its subscriber base. As shown in Figure 12.2, the VV21-WM consists of three sub-modules, namely, Demand Predictor (DP), Supply Curve Follower (SCF), and Bid Generator (BG). First, VV21-WM uses the DP sub-module to estimate the expected demand of its subscriber base for any delivery timeslot. After that, the BG sub-module participates in PDA by placing bids to procure electricity. The SCF sub-module aids BG in determining suitable limit prices for bids that are to be placed in an auction. Failing to procure required electricity after all auction instances of PDA, a broker has to buy the remaining electricity from the balancing market at more expensive prices. Below, we briefly discuss all three sub-modules.

Demand Predictor (DP) : The DP sub-module is used to predict the expected usage of VV21's subscriber base up to 24 hours into the future by utilizing a simple weekly

moving average-based technique for each customer. The result is then multiplied by the current number of subscriptions of that customer, and the total usage for a delivery slot is then obtained by summing the predicted usage across all customers in the tariff portfolio.

Supply Curve Follower (SCF): The SCF sub-module helps in determining suitable limit prices for generating bids by learning the price curve of the GenCo, which is quadratic in nature and then following this curve through uncleared asks made available to the broker. This helps in identifying the least ask that GenCo could put for a particular auction and use it to generate bids via the BG sub-module.

Bid Generator (BG): The BG sub-module is responsible for generating multiple bids based on the outputs of DP and SCF. When the bidding time is far from the delivery time, BG generates bids with limit prices less than the predicted least ask of GenCo, hoping to capitalize on asks from smaller sellers. Closer to the delivery slot, BG generates bids with limit prices closer to or greater than the least ask of GenCo. This ensures procurement of the outstanding energy from GenCo and thereby avoids going to the balancing market.

12.3.3 Tariff Module of VidyutVanika21

As shown in Fig. 12.2, VV21-TM has three sub-modules, namely, Tariff Enhancer (TE), Tariff Designer (TD), and Tariff Health Checker (THC). The TE sub-module calculates mean tariff rates to maintain tariff market share within predefined bounds. The TD sub-module generates weekly ToU tariffs based on the mean rates suggested by TE with surcharges during peak hours. The THC sub-module periodically checks for active tariffs' profitability and subscription rates and removes loss-making and under-subscribed tariffs. Below, we briefly discuss all three sub-modules.

Tariff Enhancer (TE): TE is in charge of maintaining an appropriate level of market share in a game by improving VV21's active tariffs by considering all active tariffs present in the market by utilizing the game theoretical analysis for optimal market share. First, TE computes the cumulative market share of all its active tariffs, then depending on the scenario, TE chooses to remove its cheapest tariff and suggest a costlier tariff to reduce market share or suggests a cheaper tariff to help increase the market share. After that, it calls the TD sub-module to generate weekly ToU tariffs.

Tariff Designer (TD): TD is responsible for designing a weekly ToU tariff based on the average input price received from TE with the use of an empirically designed weekly tariff pattern indicating peak and non-peak timeslots. Based on this weekly pattern, we impose surcharges during peak hours to mitigate the effects of capacity transaction penalties.

Tariff Health Checker (THC) : THC aims to remove unhealthy tariffs from the tariff market periodically. Based on the information stored in VV21's tariff repository, tariffs that are loss-making or under-subscribed are marked as unhealthy and are removed from the market.

Both VV21-WM and VV21-TM complement each other during the games. The procurement price of electricity with some profit margin acts as a base value for the VV21-TM to design tariff contracts; in this way, VV21 always remains profitable through electricity sales. Furthermore, the production and storage customers of VV21 in the tariff market help it to adjust the electricity shortage by supplying electricity and thus save from costlier prices in the balancing market; furthermore, they help VV21 to get profitable in the balancing market by helping the market to recover from a deficit of electricity by supplying needed electricity to market. Thus, the coordination between both modules of VV21 paves

the way for its success in the PowerTAC tournament. Below, we present a detailed analysis of PowerTAC2021 finals and highlight VV21’s performance.

12.3.4 Performance Analysis

In this section, we present the results of the PowerTAC2021 finals and showcase the performance of VV21. As stated previously, the 2021 edition of the tournament had 7 finalists; the other six competing brokers were TUC_TAC, CrocodileAgent2020 (Crocodile), IS3, COLDPOWER2021 (ColdPower), UTA_PTA2021 (UTA_PTA) and Mertacor2021 (Mertacor). The tournament had a total of 386 games played across 3,5 and 7-Player configurations in three real-world locations: Denver, Minneapolis, and Phoenix. Each broker was allocated 50 7-Player, 90 5-Player, and 90 3-Player games.

Leaderboard Analysis: We first present the leaderboard of the PowerTAC2021 tournament finals. Due to the difficulty of presenting the full leaderboard as a single table, we divide the leaderboard into two parts, containing Unnormalized scores [12.1](#) and Normalized scores [12.2](#). As shown in Table [12.1](#), 7 brokers contested across three different player configurations, and VV21 emerged with the highest unnormalized scores in all three configurations; thus, achieved the highest total unnormalized score that is almost double the runner-up broker. The unnormalized scores are nothing but the cumulative cash positions of each broker in the above-discussed player configurations. The total scores are the summation of the broker’s scores in those three player configurations. The number in the brackets in each cell indicates the percentage cash position with respect to the winning broker in that player configuration based on unnormalized scores. As VV21 is the winner in all three player configurations, it is assigned 100% in all three player configurations as well as in the total scores. The runner-up broker, TUC_TAC could only achieve 26%, 48%, and 55% of VV21’s scores in 7-Player, 5-Player, and 3-Player configurations, respectively; thus, only achieving 53% of VV21’s total score. The third-placed broker, CrocodileAgent2020, achieved 59% of VV21’s total score, but due to a lower normalized score, it ranked below

Table 12.1: PowerTAC2021 Leaderboard (Unnormalized)

Broker	7-Player (50)	5-Player (90)	3-Player (90)	Total (230)
VidyutVanika21	12.5M(100)	79.3M(100)	375.1M(100)	466.8M(100)
TUC_TAC	3.3M(26)	38.3M(48)	207.5M(55)	249.2M(53)
CrocodileAgent2020	-1.6M(-13)	19.6M(24)	258.3(68)	276.2(59)
IS3	2.3M(18)	13.6M(17)	65.5M(17)	81.4M(17)
COLDPOWER2021	-13.2M(-105)	-66.0M(-83)	16.3M(4)	-62.8M(-13)
UTA_PTA2021	-20.5M(-164)	-100.7M(-127)	4.4M(1)	-116.8M(-25)
Mertacor2021	-34.6M(-277)	-70.2M(-88)	8.2M(2)	-96.7M(-20)

TUC_TAC. Out of the remaining four brokers, COLDPOWER2021, UTA_PTA2021, and Mertacor2021 ended the tournament with negative scores.

Table 12.2 shows the normalized scores of each broker, which is used to determine the winner of the tournament. The normalized scores are calculated by performing the standardization (or Z-Score Normalization) individually with the help of mean and standard deviation for each player configuration. The leaderboard numbers demonstrate that VV21 was the winner of the PowerTAC2021 tournament. Here too, based on the normalized scores, VV21 was better than all the other brokers, achieving almost double the score of the second-best broker in all three configurations. Thus, our modelling of the PowerTAC game as a zero-sum game maximizing the difference between the cash position of VV21 and its opponents (refer Section 9.2), along with the ability to procure energy in wholesale and balancing markets for lower prices, helped VV21 to achieve approximately double scores than the second-placed broker in each configuration. The final score of 4.58 achieved by VV21 is the highest in the history of PowerTAC tournaments.

Table 12.2: PowerTAC2021 Leaderboard (Normalized)

Broker	7-Player (50)	5-Player (90)	3-Player (90)	Total (230)
VidyutVanika21	1.32	1.48	1.77	4.58
TUC_TAC	0.72	0.82	0.54	2.08
CrocodileAgent2020	0.38	0.52	0.91	1.81
IS3	0.64	0.42	-0.50	0.57
COLDPOWER2021	-0.38	-0.70	-0.86	-2.11
UTA_PTA2021	-0.87	-1.43	-0.95	-3.25
Mertacor2021	-1.81	-0.94	-0.92	-3.67

Wins Analysis: Figure 12.3 shows the plots for a number of 1st and 2nd ranks of each broker in the PowerTAC2021 tournament finals. Upon analyzing the plot, we notice that both VV21 and runner-up TUC_TAC won more than half of their respective games (total games played by them are 230). More precisely, VV21 achieved 1st rank in 121 games and 2nd rank in 58 games, while TUC_TAC achieved 1st rank in 132 games and 2nd rank in 42 games. In fact, TUC_TAC won slightly more games than VV21 but lost considerably in the other games, whereas VV21 managed to curtail its losses in the games where it could not win. Interestingly, the last-placed broker, Mertacor, won more games than the 3rd to 6th placed brokers; however, it ended up making towering losses when it lost games and finished in the last position. Thus, the capability to curtail losses also played a decisive role in the victory of VV21 in the 2021 tournament.

Negative Profits Analysis: To analyze the results further, we show the number of games where brokers ended up having negative profits in the PowerTAC2021 tournament finals in Figure 12.4. In 2021, VV21 ended up in negative profits in fewer games than any other broker in the tournament despite maintaining a considerable market share. More

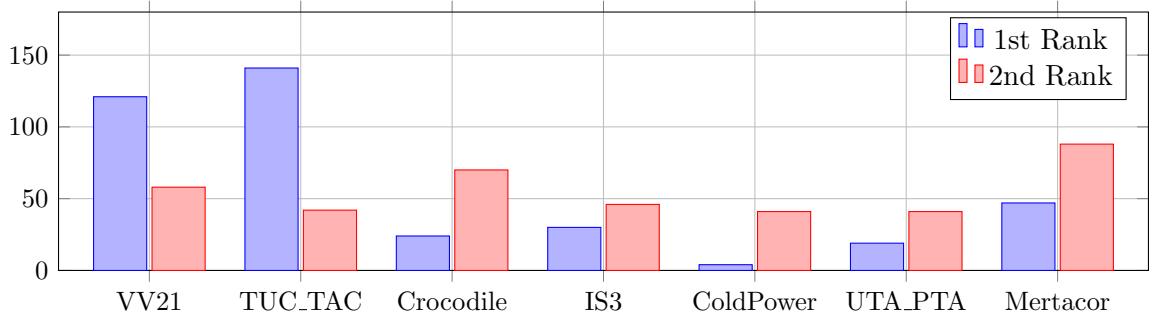


Figure 12.3: Number of 1st and 2nd ranks of each broker in PowerTAC 2021

accurately, VV21 had negative cash positions in 47 games out of the 230 games that it played. The second best broker in this metric is TUC_TAC which had 50 games with negative cash positions out of the 230 games that it played. On the other extreme, brokers COLDPOWER2021, UTA.PTA2021, and IS3 had 126, 127, and 144 games, respectively, having negative cash positions out of a total of 230 games. Thus, VV21 consistently managed to make up for its losses and rarely became non-profitable.

Market Share Analysis: As VV21's tariff strategy is concerned with maintaining an appropriate market share for each player configuration, we analyze the average market share maintained by VV21 during the tournament in Figure 12.5. Recall that the optimal market shares based on game theoretical analysis for 7-Player, 5-Player, and 3-Player are 30.0%, 38.55%, and 48%, respectively (refer Section 9.2). The average market shares maintained by VV21 during the PowerTAC2021 finals are 38.03%, 40.44%, and 55.9% in 7-Player, 5-Player, and 3-Player, respectively, which are not very far from the optimal. Thus, VV21 was able to maintain market share during the finals as expected.

However, as mentioned previously, the Phoenix region introduces unexpectedly high demands from customers. Due to this, the market ends up in deficit as GenCos are not able to supply the required electricity to fulfil extraordinary demands. In such scenarios, having more **Consumption** customers generally leads to high electricity procurement costs as the

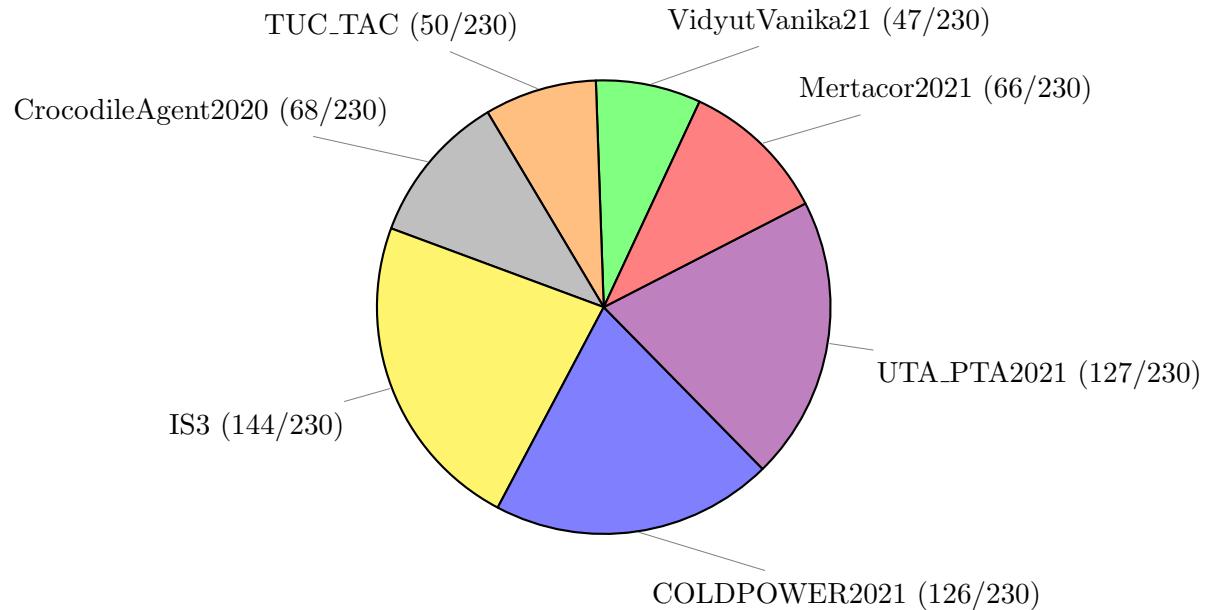


Figure 12.4: Games with Negative Cash in PowerTAC 2021

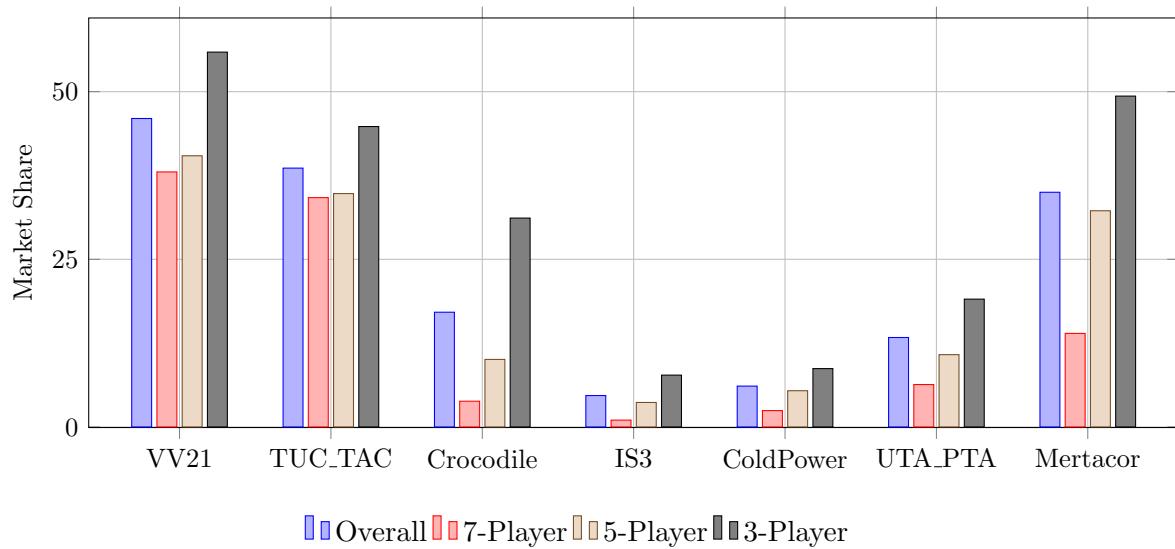


Figure 12.5: Average Market Share of Brokers in PowerTAC 2021

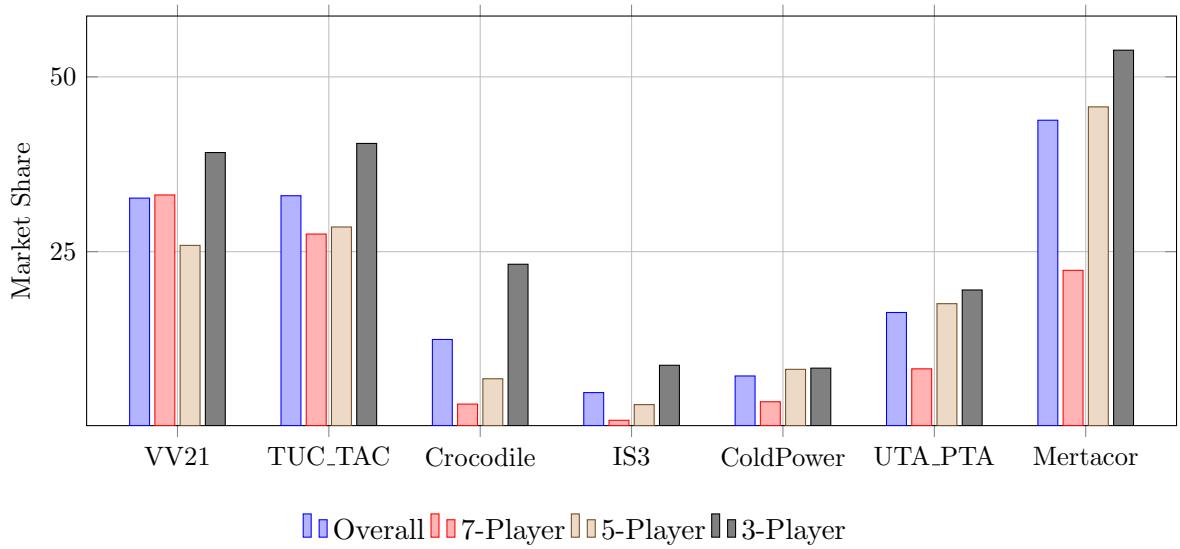


Figure 12.6: Average Market Share of Brokers in Phoenix Games in PowerTAC 2021

electricity rates are towering high; on the other hand, having *production and storage* customers may help to reduce the market deficit by supplying electricity to the market and thus gain profits. As explained, VV21 aims to create a balanced portfolio of customers with its tariff module; thus, it accommodates all types of customers, consumption, production, and storage. During Phoenix games, VV21 let off a portion of consumption customers to manage the costs and remain profitable in the games. Figure 12.6 shows the market share of all the brokers in the Phoenix games. As shown in the figure, VV21's market shares in 7-Player, 5-Player, and 3-Player games were 33.11%, 25.9%, and 39.18%, respectively, quite less than the average market share in the overall PowerTAC finals. Upon analyzing further, we note that Mertcor's market shares increased during the Phoenix games than its average market shares in the finals; having more consumption customers turned out to be the prominent reason for Mertcor's downfall in the finals as it lost dearly in the Phoenix games. Note that the total market share is dominated by the consumption type customers; thus, having a higher market share implies having higher consumption customers.

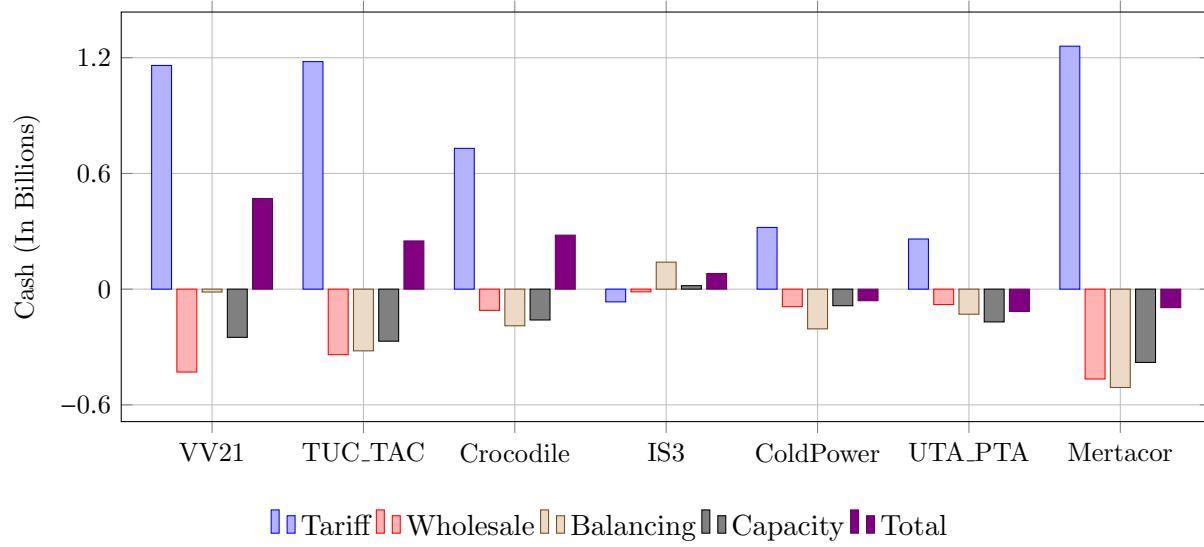


Figure 12.7: Accounting Information of Brokers in PowerTAC 2021

Accounting Analysis: Accounting analysis of the PowerTAC2021 tournament finals is presented in Fig. 12.7 that furnishes the break up of broker revenue in the tariff, wholesale, and balancing market, along with capacity transaction penalties and the final cash position of each broker. In 2021, an effective tariff strategy resulted in handsome profits in the tariff market and lower capacity transaction charges. Furthermore, a strong wholesale and tariff strategy contributed to low balancing costs. Additionally, VV21 performed the best in terms of the *income-to-cost ratio (1.67)* among the revenue-making brokers of the tournament. The runner-up broker, TUC_TAC, too, had a handsome revenue in the tariff market; however, due to high balancing costs, its revenue decreased. Remarkably, Mertacor had the highest revenue in the tariff market and still ended up in the last place in the finals. The major reason for this is the losses it incurred in the Phoenix games.

Ablation Analysis: In this section, we perform controlled ablation experiments to analyze the efficacy of each constituent module of VV21. To this end, we disable critical modules of VV21 individually to create new brokers and report the performance drop observed

Table 12.3: Ablation Analysis of VidyutVanika21

Broker	VV21_RM	VV21_WM	VV21_TD	VV21_CONS
Profit (%)	50.54	60.16	79.70	91.92

by playing 50 two-player games between full VV21 and the curtailed brokers. VV21_RM and VV21_WM are agents created by substituting the tariff and wholesale module of VV21 with the corresponding strategies of the PowerTAC sample broker. The agent VV21_TD is generated by disabling the tariff designer sub-module of VV21. To study the effect of maintaining a balanced portfolio, we consider an agent VV21_CONS that offers only consumption tariffs.

The performance drop figures are reported in Table 12.3, which is the percentage profit the curtailed broker generates compared to full VV21. VV21_RM and VV21_WM were only able to generate 50.54% and 60.16% profits, respectively, indicating the importance of the tariff and wholesale module of VV21. VV21_TD manages to achieve 79.70% profit, thus reinforcing the utility of weekly ToU tariffs with peak hour surcharges to mitigate capacity transition charges. The performance drop of VV21_CONS to 91.92% indicates the usefulness of having a balanced portfolio with production and storage customers.

12.4 VidyutVanika22 in the PowerTAC2022 Tournament

The PowerTAC2022 tournament was organized in November-December 2022. This year's tournament also had a few trial rounds to help teams test and rectify their autonomous broker's strategies. The qualifier round of PowerTAC2022 happened in November, where a total of 5 teams participated, and all 5 teams got qualifiers for the finals. This year too, VidyutVanika topped the qualifiers round.

The final of PowerTAC2021 took place from 6 to 14 December, 2022, in which all 5 qualified teams participated. A total of 340 games played between 5 brokers in 5-Player,

3-Player, and 2-Player configurations. A total of 20 5-Player, 96 3-Player, and 64 2-Player games were organized in the finals of PowerTAC2022; thus, each broker played exactly 180 games in the finals. The games happened in four different weather locations, namely, Denver, Minneapolis, Cleveland, and Phoenix; the weather location is randomly chosen at the start of the game. This year, too, during the summertime in the Phoenix region, the electricity usage of customers goes extraordinarily high, which creates a deficit of electricity, and thus, electricity prices surge to record high values. Below, we discuss the architectural details of our broker, VidyutVanika22, who participated in the PowerTAC2022 tournament.

12.4.1 Design and Overview of VidyutVanika22

As briefed in the overview, VidyutVanika22 (VV22) uses VV21’s wholesale and tariff modules in some games, while in other games, it deploys learning-based strategies in both wholesale and tariff markets of PowerTAC. Figure 12.8 shows the architecture of VidyutVanika22. The wholesale module is denoted as VV22-WM, and the tariff module is called VV22-TM. VV22-WM is responsible for energy procurement in the wholesale market, while VV22-TM handles tariff publication and revocation in the tariff market. Additionally, VV22 maintains repositories to store data received from the game server about the weather, wholesale, balancing, and tariff markets, which is then used in the decision-making process.

For its learning-based strategies, VV22-WM follows an RL-based strategy implemented using DDPG to deploy a scale-based bidding strategy to participate in PDA. The strategy is backed up by Bayesian Nash equilibrium analysis and has shown to be effective in learning the scale factors for scale-based bidding. The strategy and corresponding analysis are presented in detail in Chapter 5. The strategy is designed to procure as much electricity as possible from the wholesale market to reduce the balancing market transactions. The strategy used by VV22-TM is inherently the same as VV21-TM in terms of execution; however, VV22-TM utilizes a MAB-based strategy to design tariff contracts to achieve

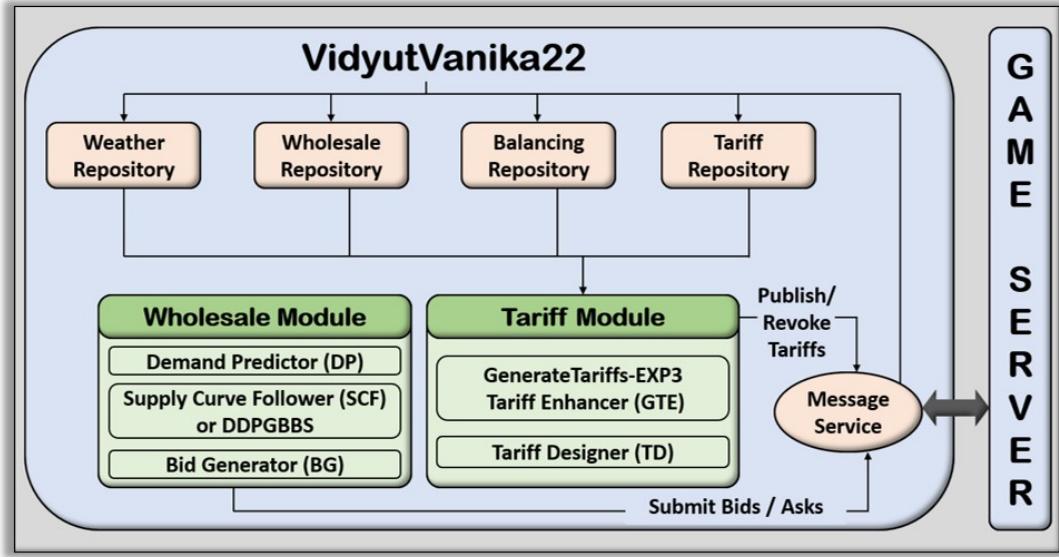


Figure 12.8: VidyutVanika22: System Architecture

and maintain optimal market share during the games, essentially replacing VV21-TM's heuristics with a learning-based method. The details of the strategy are presented in Chapter 10. VV22's tariff module, too, targets achieving a balanced portfolio consisting of customers of different types (consumption, production, and storage). Tariffs usually contain surcharges during peak hours to further mitigate the effect of capacity transaction charges. The design of VV22 helps in handling balancing market activities by reducing purchases from the balancing market, resulting in low balancing costs and, during high-demand scenarios, supplying energy via producers and storage customers in its portfolio to make up for the market deficit to generate revenue. Below, we provide a brief description of the sub-modules of VV22's learning-based strategies; details of these strategies are presented in the previous chapters.

12.4.2 Wholesale Module of VidyutVanika22

The wholesale module of VV21 (VV21-WM) aims to minimize energy procurement costs while buying needed electricity for its subscriber base. As shown in Figure 12.8, the VV22-WM consists of three sub-modules, namely, Demand Predictor (DP), Supply Curve Follower (SCF) or DDPG-based Bidding Strategy (DDPGBBS), and Bid Generator (BG). As SCF is already explained, here we only present the DDPGBBS. First, VV21-WM uses the DP sub-module to estimate the expected demand of its subscriber base for any delivery timeslot. After that, the BG sub-module participates in PDA by placing bids to procure electricity. The DDPGBBS sub-module aids BG in determining suitable limit prices for bids that are to be placed in an auction. Failing to procure required electricity after all auction instances of PDA, a broker has to buy the remaining electricity from the balancing market at more expensive prices. Below, we briefly discuss DDPGBBS; other sub-modules are presented in Section 12.3.2.

DDPG-based Bidding Strategy (DDPGBBS) : The DDPGBBS sub-module helps in determining suitable limit prices for generating bids by learning the scale factors for its scale-based bidding strategy. To determine the scale factor, DDPGBBS trains and updates a DDPG-based network as shown in Section 5.4.3. Finally, VV22-WM generates bids via the BG sub-module using the limit prices suggested by the DDPGBBS sub-module.

12.4.3 Tariff Module of VidyutVanika22

As shown in Fig. 12.8, VV22-TM has two sub-modules, namely, Generate-Tariffs-EXP3 Tariff Enhancer (GTE) and Tariff Designer (TD). The GTE sub-module calculates mean tariff rates to maintain tariff market share within predefined bounds, and then the TD sub-module generates weekly ToU tariffs based on the mean rates suggested by GTE with surcharges during peak hours. Below, we briefly discuss the GTE sub-module; the TD

sub-module is presented in Section 12.3.3.

Generate-Tariffs-EXP3 Tariff Enhancer (GTE): GTE is in charge of maintaining an appropriate level of market share in a game by improving VV22’s active tariffs by considering all active tariffs present in the market by utilizing the game theoretical analysis for optimal market share. First, using the MAB-based technique, GTE formulates the tariff generation problem as a learning problem; then, using the EXP3 10.3 algorithm, it solves the MDP and determines an appropriate rate value. After that, it calls the TD sub-module to generate weekly ToU tariffs.

12.4.4 Performance Analysis

In this section, we present the results of the PowerTAC2022 finals, along with focusing on the performance of VV22. As stated previously, the PowerTAC2022 finals had only 5 teams; however, these brokers are some of the most consistent, and few of them are past winners of the tournaments. These four competing brokers were TUC_TAC22, Mertacor2022 (Mertacor), IS3, and COLDPOWER2022 (ColdPower). The tournament had a total of 340 games played across 5,3, and 2-Player configurations in four real-world locations: Denver, Minneapolis, Cleveland, and Phoenix. Each broker was allocated 20 5-Player, 96 3-Player, and 64 2-Player games, each broker playing a total of 180 games.

Leaderboard Analysis: Similar to PowerTAC2021 analysis, we first present the leaderboard of the PowerTAC2022 tournament finals as two separate tables, containing Unnormalized scores 12.4 and Normalized scores 12.5. As shown in Table 12.4, 5 brokers contested across three different player configurations, and VV22 continued its dominant performance in the second consecutive year and emerged with the highest unnormalized scores in all three configurations; thus, it achieved the highest total unnormalized score that is almost 1.5 times the runner-up broker. Similar to the PowerTAC2021 leaderboard, the number in the brackets in each cell indicates the percentage cash position with respect to the winning

Table 12.4: PowerTAC2022 Leaderboard (Unnormalized)

Broker	5-Player (20)	3-Player (96)	2-Player (64)	Total (180)
VidyutVanika22	10.4M(100)	173.2M(100)	218.9M(100)	402.6M(100)
TUC_TAC22	0.32M(3)	96.6M(56)	158.2M(72)	255.1M(63)
Mertacor2022	-9.56M(-187)	-22.4M(-13)	192.4(88)	150.5(37)
IS3	-16.1M(-155)	15.8M(9)	-7.6M(-3)	-7.9M(-2)
COLDPOWER2022	-57.7M(-553)	-245.4M(-142)	-116.1M(-53)	-419.2M(-104)

broker in that player configuration based on unnormalized scores. VV22, being the winner in all three configurations, is assigned 100% in all three player configurations as well as in the total scores. The runner-up broker, TUC_TAC could only achieve 3%, 56%, and 72% of VV22’s scores in 5-Player, 3-Player, and 2-Player configurations, respectively; thus, only achieving 63% of VV22’s total score. The third-placed broker, Mertacor2022, achieved 37% of VV22’s total score. The remaining two brokers, IS3 and COLDPOWER2022, ended the tournament with negative scores.

Table 12.5 shows the normalized scores of each broker, demonstrating that VV22 was the winner of the PowerTAC2022 tournament. Here too, based on the normalized scores, VV22 was better than all the other brokers, achieving almost double the score of the second-best broker in all three configurations. In this tournament, too, our modelling of the PowerTAC game as a zero-sum game maximizing the difference between the cash position of VV22 and its opponents helped VV22 to continue its prevalent performance and achieve the final score of 3.36.

Wins Analysis: Figure 12.9 shows the plots for a number of 1st and 2nd ranks of each broker in the PowerTAC2022 tournament finals. In this tournament, VV22 was vastly superior to other brokers and finished in the top 2 in 166 games out of the 180 games it

Table 12.5: PowerTAC2022 Leaderboard (Normalized)

Broker	5-Player (20)	3-Player (96)	2-Player (64)	Total (180)
VidyutVanika22	1.16	1.20	1.00	3.36
TUC_TAC22	0.72	0.66	0.53	1.91
Mertacor2022	-0.13	-0.18	0.80	0.49
IS3	0.02	0.09	-0.75	-0.64
COLDPOWER2022	-1.77	-1.76	-1.59	-5.11

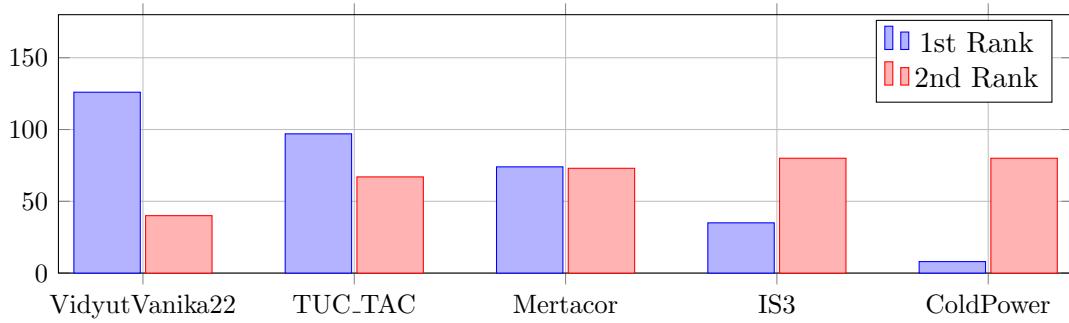


Figure 12.9: Number of 1st and 2nd ranks of each broker in PowerTAC 2022

played, out of which VV22 was the winner in 120 games (67% winning rate). Whereas the runner-up TUC-TAC won only 97 games out of the 180 games it played.

Negative Profits Analysis: We show the number of games where brokers ended up having negative profits in the PowerTAC2022 tournament finals in Figure 12.10. Similar to last year’s tournament, in 2022 too, VV22 ended up in negative profits in fewer games than any other broker in the tournament despite maintaining a considerable market share. More accurately, VV22 had negative cash positions in 20 games out of the 180 games that it played, far less than the second-best broker TUC-TAC which had 32 games with negative cash positions. On the other extreme, brokers like IS3 and COLDPOWER2021 had 77

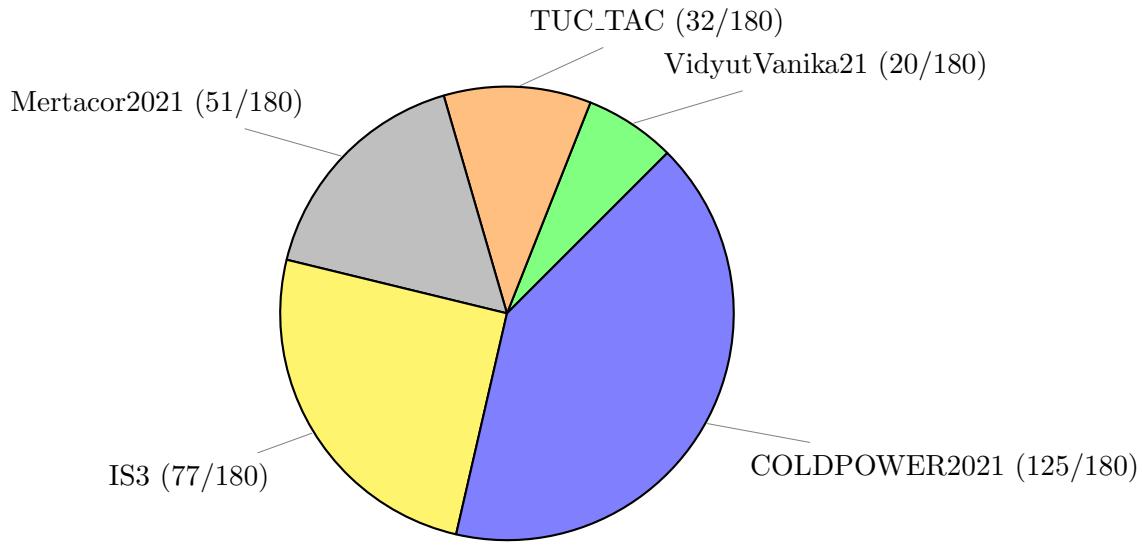


Figure 12.10: Games with Negative Cash in PowerTAC 2022

and 125 games, respectively, having negative cash positions out of a total of 180 games. Thus, in the second consecutive tournament, VV22 consistently managed to make up for its losses and rarely became non-profitable.

Market Share Analysis: Consistent with our belief of maintaining an appropriate market share during the games, VV22's tariff strategy, too, maintains an appropriate market share for each player configuration. We analyze the average market share maintained by VV22 during the tournament in Figure 12.11. Recall that the optimal market shares based on game theoretical analysis for 5-Player, 3-Player, and 2-Player are 38.55%, 48%, and 60%, respectively (refer Section 9.2). The average market shares maintained by VV22 during the PowerTAC2022 finals are 29.50%, 43.80%, and 54.80% in 5-Player, 3-Player, and 2-Player, respectively, which are close to the optimal. Thus, VV22 was able to maintain market share during the finals, just like VV21.

Here, we also carry out a similar market share analysis for the Phoenix region. Similar to VV21, VV22 also lets off a portion of consumption customers to manage the costs and

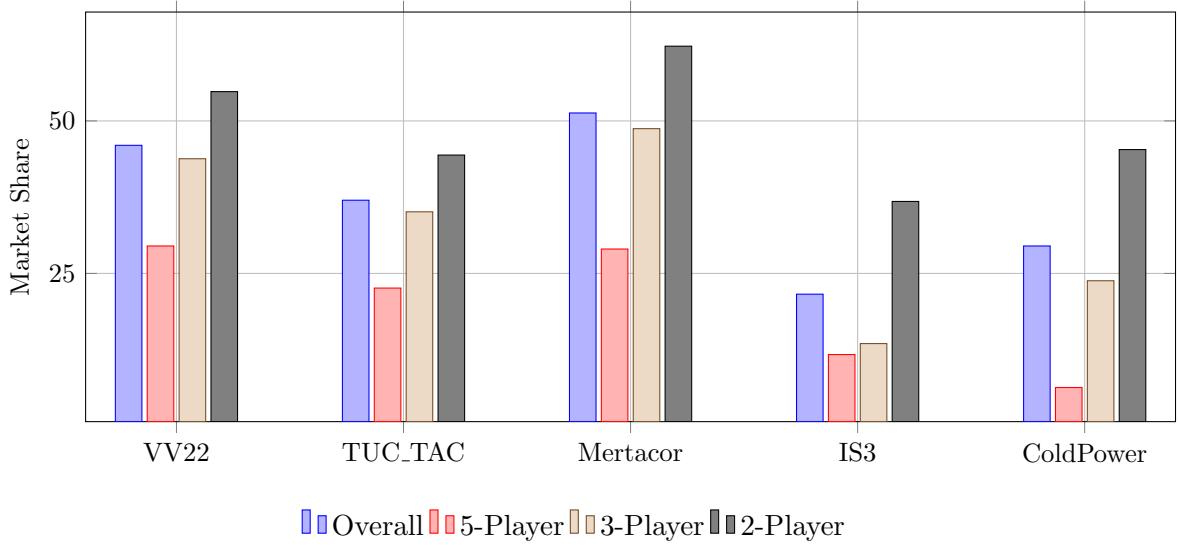


Figure 12.11: Average Market Share of Brokers in PowerTAC 2022

remain profitable in the games. Figure 12.12 shows the market share of all the brokers in the Phoenix games. As shown in the figure, VV22's market shares in 5-Player, 3-Player, and 2-Player games were 26.60%, 37.8%, and 42.00%, respectively, quite less than the average market share in the overall PowerTAC finals. Here, too, brokers who had higher market shares during the Phoenix games than their average market shares in the finals ended up with high losses in the tournament.

Accounting Analysis: Accounting analysis of the PowerTAC2022 tournament finals is presented in Fig. 12.13 that shows the break up of broker revenue in the tariff, wholesale, and balancing market, along with capacity transaction penalties and the final cash position of each broker. The 2022 accounting analysis shows a similar trend as the 2021 analysis. In 2022 as well, VV22 earned handsome profits in the tariff market and lower capacity transaction charges, along with balancing costs and wholesale costs. The runner-up broker, TUC_TAC, too, had a handsome revenue in the tariff market; but again, due to high balancing costs, its revenue decreased. Similar to last year, Mertacor had the highest

revenue in the tariff market, but due to high balancing and capacity transactions, it ended up in third place in the finals.

12.5 Guidelines for Autonomous Agent Development for Smart Grids

We have been participating in the PowerTAC tournaments for five years. During this period, we attempted many different strategies and techniques. We have already discussed the successful strategies; however, some strategies/techniques did not result in satisfactory performance, and we learned those are unsuitable for the PowerTAC environment. Below, we highlight some of the crucial lessons we learned that could be useful for broker development in the future.

The first lesson relates to usage prediction during a game in PowerTAC simulation. To achieve higher prediction accuracy for customer usage and net demand prediction, which is a time series prediction task, we endeavoured various *offline* learning techniques ranging from statistical analysis (i.e., ARIMA) to machine learning (i.e., FFN, RNN like LSTM and GRU). We observed that our models did not consistently outperform the sample broker's exponential average prediction method during the tournament games, even after training on enough data points and utilizing the best-known architectures. The lesson here is that offline prediction is unsuitable for usage prediction in the PowerTAC simulator because of the simulator's inbuilt randomness and other exogenous elements like weather. In PowerTAC, there is inbuilt randomness to model customer behavior at the start of the game; additionally, the weather location and the season are also selected randomly at the start, leading to high randomness from game to game. Thus, we need to collect as many data classes as possible for offline training. It would be tough for a prediction model to learn from the data having high variation in the usage patterns. Moreover, even if the model learns, it would learn an average response based on all collected data, which may

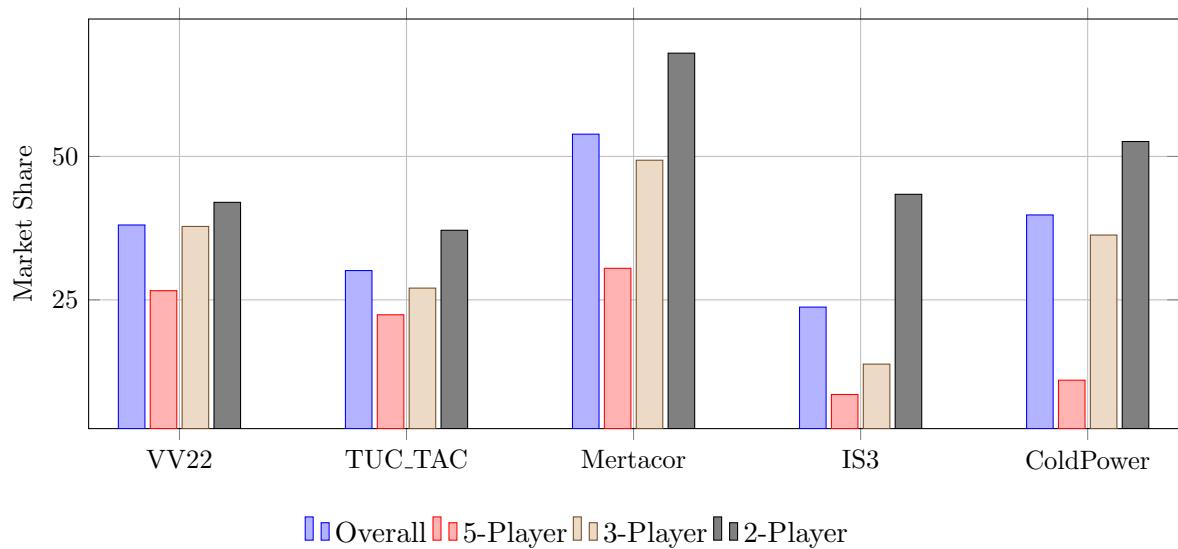


Figure 12.12: Average Market Share of Brokers in Phoenix Games in PowerTAC 2022

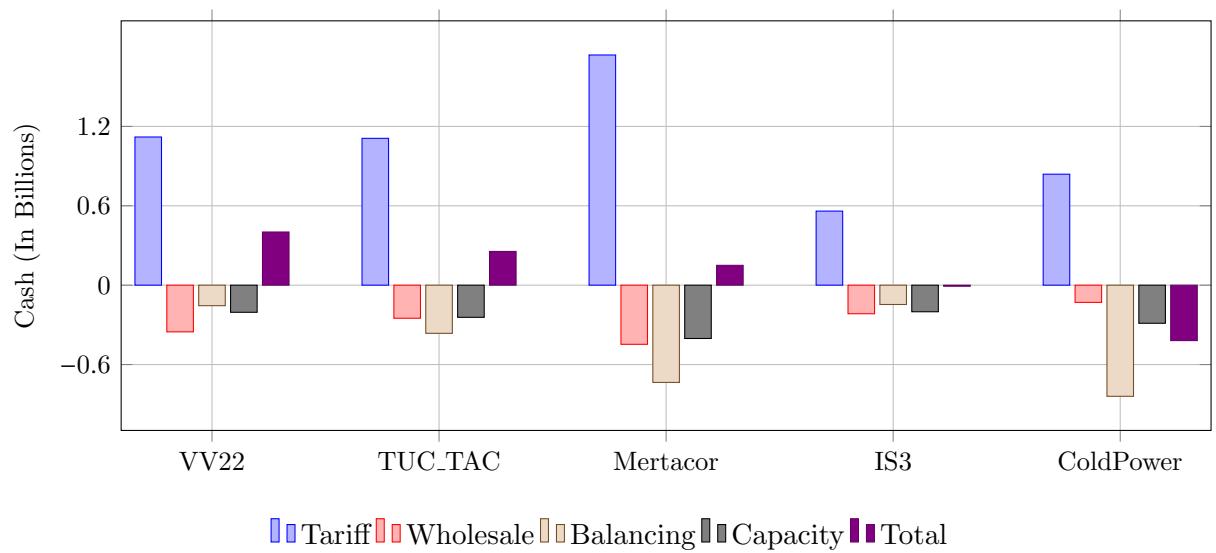


Figure 12.13: Accounting Information of Brokers in PowerTAC 2022

not be suitable for usage prediction in PowerTAC as even a small imbalance leads to high balancing penalties. The more accurate way is to make online predictions within the game and not rely on previous game data.

Another important learning we came across is the design of tariff strategy. For designing any learning-based strategy (i.e., RL), one needs to collect enough data points, possibly millions, to train the model over the maximum possible state space. However, the problem occurs while generating data points for any tariff strategy. To design any tariff strategy, one needs to select an appropriate state space containing information about one's own current tariffs, opponents' tariffs, market situation, etc. Thus, the state space has to be sufficiently large, and we need millions of data points, which is problematic as tariff usage data is generated once in six timeslots during the game. Thus, there are around 240 data points in a game, which implies we need to play thousands of games to generate enough data for training, which is highly unlikely to execute in the PowerTAC.

12.6 Summary

This chapter shows the system architectures of the broker VidyutVanika that participated in the PowerTAC tournaments in 2021 and 2022. We show how the strategies presented in the previous chapters are combined to design a fully autonomous broker for a smart grid system. Furthermore, we show a detailed analysis of VidyutVanika's performance in both tournaments, where VidyutVanika emerged as a stand-out winner. Finally, we discuss guidelines for designing autonomous agents for smart grid systems that could be useful for an aspiring smart grid researcher.

Chapter 13

Conclusion and Future Work

Conclusion

In this work, we showcased the design of autonomous brokers for smart grid systems from a theoretical and design point of view. The initial chapters focused on developing the necessary background to understand the novelty of the work presented in this thesis. We divided our contributions into two parts: wholesale and tariff strategies.

Part A (Wholesale Market) – Designing Bidding Strategies in Wholesale Market. The bidding strategies proposed in the literature for PDA are either too complex or too slow for real-time systems. To remedy this, we proposed two novel bidding strategies for PDAs. Both the proposed strategies are easy to deploy and real-time and aim to minimize electricity procurement costs by planning for current and future auctions. The first strategy takes the theoretical Nash equilibrium analysis as a foundation to design a real-time scale-based strategy to place bids in PDAs. The second strategy uses market

information to come up with the bids; it models the supply curve of the seller to figure out appropriate bids for the auction.

The contributions in the tariff market are further divided into two parts: tariff strategy generation and peak demand reduction strategies.

Part B₁ (Tariff Market) – Designing Tariff Generation Strategies in Tariff Market.

In the tariff market, brokers need to be dynamic and periodically examine and improve tariff contracts in order to be competitive. We proposed two such novel tariff-generation strategies in this work. The fundamental idea in both strategies is the game-theoretical analysis of an appropriate market share for the broker. Both strategies use this analysis as a base to achieve and maintain appropriate market shares during the games, either using heuristics or learning-based methods. The first strategy uses intelligent heuristics to achieve the goal, whereas the second strategy makes use of MAB literature to design a learnable tariff generation strategy to achieve the goal.

Part B₂ (Tariff Market) – Designing Tariff-based Demand Response for Peak Reduction in Smart Grid.

Peak demand penalties are one of the major reasons for brokers' losses in the market; peak demands destabilise the grids as well. We proposed a novel strategy based on demand response to incentivize customers appropriately to reduce peak demands. Our strategy first learns the customers' response to DR incentives and proposes a model for customers' reduction probability, which is parameterized by customers' reduction rates. Then, we proposed two algorithms to allocate the available discounts optimally for the two cases when we know the customers' reduction rates and when we do not know, respectively.

Finally, we combined all the strategies presented above in the last chapter to showcase the design of our autonomous brokers, VidyutVanika21 and VidyutVanika22. We described the architectures of both brokers, followed by a rigorous analysis of the PowerTAC2021 and

PowerTAC2022 tournaments to compare the performance of our brokers against various opponents. We, team VidyutVanika, became the champions of the final two editions of the PowerTAC tournaments; thus, based on our knowledge, we shared a guideline for autonomous agent development in the future.

Future Work

In Wholesale and Tariff Markets: In the future, one may aim to design complete learning-based strategies by incorporating more complex information about the market in the tariff market and supply curve details in the wholesale market. Such learning-based strategies for tariff and wholesale markets would adapt from game to game and play appropriately against different brokers and player configurations. Along similar lines, one extension of VV's tariff strategy may possibly be to learn equilibrium market share online depending on player configuration and the nature of players, which will lead to maintaining appropriate market shares against stronger and weaker opponents during the game.

For Peak Reduction: In our work, we performed an empirical analysis to showcase that the regret of our proposed algorithms is sub-linear. One may attempt to prove the sub-linear regret of the algorithms theoretically. The proposed linear search method searches for the optimal reduction rates linearly in an interval; one may explore more ways to optimize the search time of the algorithm. In our work, the broker gives discounts to customers in the form of tariffs, but we do not track the actual discount in the peak revenues of the broker. One may study the relation between the given discount and the actual discount observed in the market and use it to improve the proposed strategy.

In Balancing Market: Apart from the wholesale and tariff markets, there has not been much work done in the balancing market. VV21 exploits the balancing market and earns handsome profits in games with unusually high demands but does not earn much profit

from the balancing market in games with standard demands. A bidding strategy for the balancing market can be designed by incorporating an accurate predictor of market surplus and deficit. However, such a bidding strategy in the balancing market has to be integrated with the wholesale bidding strategy. This would enable the wholesale strategy to buy less/more during the wholesale auctions based on the predicted market surplus/deficit. At the same time, the remaining/extraneous energy can be traded in the balancing market.

Prediction Models: We need a reliable prediction model for any strategic decisions in PowerTAC markets, i.e., customer usage predictor and net demand predictor. A more robust prediction model incorporating the variability of usage patterns based on offered tariffs (basically a prediction model integrated with demand response) would be more accurate for dynamic usage patterns.

Bibliography

- [1] E. Mylonas, N. Tzanis, M. Birbas, and A. Birbas. An automatic design framework for real-time power system simulators supporting smart grid applications, 2020.
- [2] Y. Narahari. *Game Theory and Mechanism Design*, volume 4, chapter 14, pages 205–222. World Scientific Publishing Co. Pte. Ltd., 2014.
- [3] D. Silver. Lecture series on introduction to reinforcement learning with david silver, May 2015.
- [4] M. Mohiuddin, I. Boiko, R. Azzam, and Y. Zweiri. Closed-loop stability analysis of deep reinforcement learning controlled systems with experimental validation. *IET Control Theory and Applications*, 18:n/a–n/a, 06 2024.
- [5] E. Editors. Overview of Smart Grid Technology And Its Operation and Application . <https://www.elprocus.com/overview-smart-grid-technology-operation-application-existing-power-system/>. [Online; accessed 2-July-2023].
- [6] W. Ketter, J. Collins, and M. de Weerdt. The 2020 power trading agent competition (march 30, 2020). *ERIM Report Series Reference No. 2020-002*, <http://dx.doi.org/10.2139/ssrn.3564107>, 2020.
- [7] A. Haleem, M. Javaid, M. A. Qadri, R. P. Singh, and R. Suman. Artificial intelligence (ai) applications for marketing: A literature-based study. *International Journal of Intelligent Networks*, 2022.
- [8] A. Takyar. AI Use Cases & Applications across Major Industries . <https://www.leewayhertz.com/ai-use-cases-and-applications/>. [Online; accessed 2-July-2023].
- [9] E. Presswire. AI Market Share. <https://menafn.com/1104686471/AI-Edge-Computing-Market-Top-Key-Players-Industry-Growth-Analysis-Forecast-2030>, 2020. [Online; accessed 2-July-2023].

- [10] S. Disbrey and I. Zeier. Nord Pool AS Anual Report, 2023. [Online; accessed 16-January-2024].
- [11] I. S. Bayram, M. Z. Shakir, M. Abdallah, and K. Qaraqe. A survey on energy trading in smart grid. In *2014 IEEE Global Conference on Signal and Information Processing*, pages 258–262, 2014.
- [12] I. Vetsikas and N. Jennings. Bidding strategies for realistic multi-unit sealed-bid auctions. *Autonomous Agents and Multi-Agent Systems*, 21, 01 2008.
- [13] B. Shi, E. Gerdin, P. Vytelingum, and N. Jennings. An equilibrium analysis of competing double auction marketplaces using fictitious play. In *19th European Conference on Artificial Intelligence (ECAI) (01/08/10)*, pages 575–580, August 2010.
- [14] K. Chatterjee and W. Samuelson. Bargaining under incomplete information. In *Operations Research*, volume 31, 1983.
- [15] Y. Narahari. *Game Theory and Mechanism Design*, volume 4, pages 205–222. World Scientific Publishing Co. Pte. Ltd., 2014.
- [16] Y. Narahari. *Game Theory and Mechanism Design*, volume 4, chapter 20, pages 303–307. World Scientific Publishing Co. Pte. Ltd., 2014.
- [17] J. Kalagnanam and D. Parkes. Auctions, bidding, and exchange design. In: *Handbook of Quantitative Supply Chain Analysis: Modeling in the E-Business Era*. Ed. by D. SimchiLevi, S.D. Wu, and Z.J. Shen, 2005.
- [18] R. P. McAfee and J. McMillan. Auctions and bidding. *Journal of Economic Literature*, 25(2):699–738, 1987.
- [19] P. Milgrom. Auctions and bidding: A primer. *Journal of Economic Perspectives*, 3(3):3–22, 1989.
- [20] E. Wolfstetter. Auctions: An introduction. *Economic Surveys*, 10:367–421, 1996.
- [21] W. Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of Finance*, 16 (1):8–37, 1961.
- [22] J.-J. Laffont, H. Ossard, and Q. Vuong. Econometrics of first-price auctions. *Econometrica*, 63(4):953–980, 1995.
- [23] E. Clarke. Multipart pricing of public goods. *Public Choice*, 11 (1):17–33, 1971.
- [24] T. Groves. Incentives in teams. *Econometrica*, 41 (4):617–631, 1973.

- [25] M. A. Satterthwaite and S. R. Williams. Bilateral trade with the sealed bid k-double auctions: Existence and efficiency. In *Journal of Economic Theory*, volume 48, pages 107–133. Multidisciplinary Digital Publishing Institute, 1989.
- [26] S. Ghosh, S. Gujar, P. Paruchuri, E. Subramanian, and S. Bhat. Bidding in Smart Grid PDAs: Theory, Analysis and Strategy. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(02):1974–1981, Apr. 2020.
- [27] S. Chandlekar, E. Subramanian, S. Bhat, P. Paruchuri, and S. Gujar. Multi-Unit Double Auctions: Equilibrium Analysis and Bidding Strategy Using DDPG in Smart-Grids. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, AAMAS ’22, page 1569–1571, 2022.
- [28] S. Ghosh, E. Subramanian, S. P. Bhat, S. Gujar, and P. Paruchuri. VidyutVanika: A Reinforcement Learning Based Broker Agent for a Power Trading Competition. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:914–921, 2019.
- [29] S. Chandlekar, B. S. Pedasingu, E. Subramanian, S. Bhat, P. Paruchuri, and S. Gujar. Vidyutvanika21: An autonomous intelligent broker for smart-grids. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 158–164, 2022.
- [30] D. Urieli and P. Stone. Tactex’13: A champion adaptive power trading agent. In *Proceedings of the Twenty Eighth AAAI Conference on Artificial Intelligence*, pages 465–471. Association for the Advancement of Artificial Intelligence, 2014.
- [31] D. Urieli and P. Stone. Autonomous electricity trading using time-of-use tariffs in a competitive market. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*. Association for the Advancement of Artificial Intelligence, 2016.
- [32] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503, 2016.
- [33] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. 2013. cite arxiv:1312.5602Comment: NIPS Deep Learning Workshop 2013.

- [34] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis. Mastering the game of go without human knowledge. *Nature*, 550:354–359, 2017.
- [35] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. A. Sallab, S. Yogamani, and P. Pérez. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 23(6):4909–4926, 2022.
- [36] S. Ergun. A study on multi-agent reinforcement learning for autonomous distribution vehicles. *Iran Journal of Computer Science*, 550:354–359, 2023.
- [37] J. Kober, J. A. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.
- [38] T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, and S. Levine. Learning to walk via deep reinforcement learning, 2019.
- [39] M. Han, Y. Tian, L. Zhang, J. Wang, and W. Pan. Reinforcement learning control of constrained dynamic systems with uniformly ultimate boundedness stability guarantee. *Automatica*, 129:109689, 2021.
- [40] M. Han, X. Zhang, L. Xu, R. May, S. Pan, and J. Wu. *A Review of Reinforcement Learning Methodologies on Control Systems for Building Energy*. Working papers in Transport, Tourism, Information Technology and Microdata Analysis. 2018.
- [41] S. Sarker, L. Jamal, S. F. Ahmed, and N. Irtisam. Robotics and artificial intelligence in healthcare during covid-19 pandemic: A systematic review. *Robotics and Autonomous Systems*, 146:103902, 2021.
- [42] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- [43] A. A. Markov. *Theory of Algorithms*. Academy of Sciences of the USSR, 1954.
- [44] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons, Inc., USA, 1st edition, 1994.
- [45] R. E. Bellman. A markov decision process. pages 679–684. *Journal of Mathematical Mechanics*, 1957.
- [46] R. E. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, 1957.
- [47] R. Howard. *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, MA, 1960.

- [48] M. L. Puterman and M. C. Shin. Modified policy iteration algorithms for discounted markov decision problems. volume 24, page 1127–1137. Management Science, 1978.
- [49] D. Michie and R. A. Chambers. Boxes: An experiment in adaptive control. in e. dale and d. michie (eds.). page 137–152. Machine Intelligence 2, Oliver and Boyd, Edinburgh, 1968.
- [50] K. S. Narendra and R. M. Wheeler. Decentralized learning in finite markov chains. volume 6, page 519–526. IEEE Transactions on Automatic Control, AC31, 1986.
- [51] G. A. Rummery and M. Niranjan. On-line q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166. Engineering Department, Cambridge University, 1994.
- [52] R. S. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. in d. s. touretzky, m. c. mozer and m. e. hasselmo (eds.). page 1038–1044. MIT Press, Cambridge, MA., 1996.
- [53] C. J. C. H. Watkins. *Learning from Delayed Rewards*. Ph.D. thesis, Cambridge University, 1989.
- [54] C. J. C. H. Watkins and P. Dayan. Q-learning. volume 8, page 279–292. Machine Learning, 1992.
- [55] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2016.
- [56] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. A. Riedmiller. Deterministic policy gradient algorithms. In *ICML*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 387–395. JMLR.org, 2014.
- [57] L. Kocsis and C. Szepesvári. Bandit-based monte-carlo planning. In *Machine Learning: ECML 2006*, volume 4212 of *Lecture Notes in Computer Science*. Springer, Berlin, Heidelberg, 2006.
- [58] A. G. Barto, R. S. Sutton, and P. S. Brouwer. Associative search network: A reinforcement learning associative memory. volume 40, page 201–211. Biological Cybernetics, 1981.
- [59] J. Pearl. Heuristics: Intelligent search strategies for computer problem solving. Addison-Wesley, Reading, MA, 1984.
- [60] I. H. Witten. The apparent conflict between estimation and control—a survey of the two-armed problem. volume 301, page 161–189. Journal of the Franklin Institute, 1976.
- [61] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. volume 47(2-3), pages 235–256. Machine learning, 2002.

- [62] W. Ketter, J. Collins, and P. Reddy. Power tac: A competitive economic simulation of the smart grid. *Energy Economics*, 39:262–270, 2013.
- [63] W. Ketter, M. Peters, J. Collins, and A. Gupta. A multiagent competitive gaming platform to address societal challenges. *MIS Q.*, 40, jun 2016.
- [64] W. Ketter, M. Peters, J. Collins, and A. Gupta. Competitive benchmarking: An is research approach to address wicked problems with big data and analytics. *MIS Quarterly*, 40:1057–1080, 04 2016.
- [65] R. Myerson and M. A. Satterthwaite. Efficient mechanisms for bilateral trading. *Journal of Economic Theory*, 29(2):265–281, 1983.
- [66] R. B. Myerson. Optimal auction design. *Mathematics of Operations Research*, 6(1):58–73, 1981.
- [67] S. R. Williams. Efficient performance in two agent bargaining. *Journal of Economic Theory*, 41(1):154–172, 1987.
- [68] R. Wilson. Incentive efficiency of double auctions. *Econometrica*, 53(5):1101–1115, 1985.
- [69] W. Leininger, P. Linhart, and R. Radner. Equilibria of the sealed-bid mechanism for bargaining with incomplete information. *Journal of Economic Theory*, 48(1):63–106, 1989.
- [70] I. A. Vetsikas. Equilibrium strategies for multi-unit sealed-bid auctions with multi-unit demand bidders. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-Agent Systems*, AAMAS ’14, page 1053–1060. International Foundation for Autonomous Agents and Multiagent Systems, 2014.
- [71] J. F. Nash. The bargaining problem. *Econometrica*, 18(2):155–162, 1950.
- [72] J. C. Harsanyi. Approaches to the bargaining problem before and after the theory of games: A critical discussion of zeuthen’s, hicks’, and nash’s theories. *Econometrica*, 24(2):144–157, 1956.
- [73] T. C. Schelling. The strategy of conflict. *Cambridge and London: Harvard University Press*, 1960.
- [74] J. Cross. The economics of bargaining. new york, london, basic books. *Louvain Economic Review*, 37(5):601–601, 1971.
- [75] A. E. Roth. Axiomatic models of bargaining. *Lecture Notes in Economics and Mathematical Systems*, 170, 1979.
- [76] R. L. Butterworth. Bargaining: Formal theories of negotiation. edited by oran r. young. *American Political Science Review*, 72(2):661–662, 1978.

- [77] D. Urieli and P. Stone. Tactex'13: A champion adaptive power trading agent. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2014.
- [78] G. Tesauro and J. L. Bredin. Strategic sequential bidding in auctions using dynamic programming. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 2*, AAMAS '02, page 591–598, New York, NY, USA, 2002. Association for Computing Machinery.
- [79] R. Kuate, M. He, and M. Chli. An intelligent broker agent for energy trading: An mdp approach. In *Proceedings of the 2013 International Joint Conference on Artificial Intelligence*, pages 234–240, 2013.
- [80] M. M. P. Chowdhury, C. Kiekintveld, T. C. Son, and W. Yeoh. Bidding strategy for periodic double auctions using monte carlo tree search. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, page 1897–1899. International Foundation for Autonomous Agents and Multiagent Systems, 2018.
- [81] M. M. P. Chowdhury, C. Kiekintveld, S. Tran, and W. Yeoh. Bidding in periodic double auctions using heuristics and dynamic monte carlo tree search. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 166–172. International Joint Conferences on Artificial Intelligence Organization, 7 2018.
- [82] M. M. P. Chowdhury. Predicting prices in the powertac wholesale energy market. In *Proceedings of the AAAI Conference of Artificial Intelligence (AAAI)*, AAAI'16, pages 4204–4205, 2016.
- [83] T. Yee, V. Lisy, and M. Bowling. Monte carlo tree search in continuous action spaces with execution uncertainty. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, page 690–696. AAAI Press, 2016.
- [84] S. Bubeck, R. Munos, G. Stoltz, and C. Szepesvári. Online optimization in x-armed bandits. In *Proceedings of the Neural Information Processing Systems (NIPS)*, pages 201–208, 2008.
- [85] C. Mansley, A. Weinstein, and M. Littman. Sample-based planning for continuous action markov decision processes. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, volume 21, pages 335–338, 2011.
- [86] A. Weinstein and M. Littman. Bandit-based planning and learning in continuous-action markov decision processes. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, 2012.

- [87] A. Couetoux, J.-B. Hoock, N. Sokolovska, O. Teytaud, and N. Bonnard. Continuous upper confidence trees. In *Proceedings of the 5th International Conference on Learning and Intelligent Optimization (LION)*, 2011.
- [88] A. Couetoux, M. Milone, M. Brendel, H. Doghmen, M. Sebag, and O. Teytaud. Continuous rapid action value estimates. In *Proceedings of the 3rd Asian Conference on Machine Learning (ACML)*, volume 20 of *JMLR*, pages 19–31, Taoyuan, Taiwan, Province de Chine, 2011.
- [89] S. Gelly and D. Silver. Monte-carlo tree search and rapid action value estimation in computer go. *Artificial Intelligence*, 175(11):1856–1875, 2011.
- [90] T. Yee, V. Lisylis, and M. Bowling. Monte carlo tree search in continuous action spaces with execution uncertainty. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 690–696, 2016.
- [91] A. Couetoux. *Monte Carlo Tree Search for Continuous and Stochastic Sequential Decision Making Problems*. Phd thesis, Université Paris Sud - Paris XI, 2013. Data Structures and Algorithms [cs.DS].
- [92] A. Pacaud, C. Marceau, and A. Baranes. Monte carlo tree search bidding strategy for simultaneous ascending auctions. In *20th International Symposium on Modeling and Optimization in Mobile, Ad hoc, and Wireless Networks (WiOpt)*. IEEE, 2022.
- [93] A. Pacaud. *Bidding Efficiently in Simultaneous Ascending Auctions Using Monte Carlo Tree Search*. PhD thesis, Institut Polytechnique de Paris, 2024.
- [94] C. Buron, G. Zahia, and S. Daher. Mcts-based automated negotiation agent. In *PRIMA 2019: Principles and Practice of Multi-Agent Systems*, pages 22nd International Conference, Turin, Italy, October 28–31, 2019, Proceedings. Springer International Publishing, 2019.
- [95] W. Shen, B. Peng, H. Liu, M. Zhang, R. Qian, Y. Hong, Z. Guo, Z. Ding, P. Lu, and P. Tang. Reinforcement mechanism design: With applications to dynamic pricing in sponsored search auctions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 2020.
- [96] S. Orfanoudakis, S. Kontos, C. Akasiadis, and G. Chalkiadakis. Aiming for half gets you to the top: Winning powertac 2020. In *EUMAS*, pages 144–159, 2021.
- [97] D. Cliff. *Minimal-Intelligence Agents for Bargaining Behaviors in Market-Based Environments*. Technical Report HPL-97-91, Hewlett Packard Labs., 1997.
- [98] I. Buljevic, I. Pranjic, M. Mijic, J. Babic, A. Petric, and V. Podobnik. The crocodileagent 2012: Reaching agreements in a simulation of a smart grid wholesale market. In *Proceedings*

- of the first international conference on agreement technologies. CEUR-WS.org*, page 111–112, 2012.
- [99] S. Matetic, J. Babic, M. Matijas, A. Petric, and V. Podobnik. The crocodile-agent 2012: Negotiating agreements in smart grid tariff market. In *Proceedings of the first international conference on agreement technologies. CEUR-WS.org*, pages 203–204, 2012.
 - [100] G. Demijan, V. Hrvoje, B. Jurica, and P. Vedran. Crocodileagent 2018: Robust agent-based mechanisms for power trading in competitive environments. In *Computer Science and Information Systems*, 2018.
 - [101] T. G. Diamantopoulos, A. L. Symeonidis, and A. C. Chrysopoulos. Designing robust strategies for continuous trading in contemporary power markets. In *In E. David, C. Kiekintveld, V. Robu, O. Shehory, and S. Stein (Eds.). Agent-mediated Electronic Commerce. Designing Trading Strategies and Mechanisms for Electronic Markets*, page 30–44. Switzerland: Springer International Publishing, 2013.
 - [102] E. Ntagka, A. Chrysopoulos, and P. A. Mitkas. Designing tariffs in a competitive energy market using particle swarm optimization techniques. In *In S. Ceppi, E. David, V. Podobnik, V. Robu, O. Shehory, S. Stein, & I. A. Vetsikas (Eds.). Agent-mediated Electronic Commerce. Designing Trading Strategies and Mechanisms for Electronic Markets*, page 129–143. Cham: Springer International Publishing, 2014.
 - [103] S. Özdemir and R. Unland. Agentude: The success story of the power tac 2014’s champion. In *Workshop on Agent-Mediated Electronic Commerce and Trading Agent Design and Analysis (AMEC/TADA 2015)*, 2015.
 - [104] T. Urban and W. Conen. Maxon16: A successful power tac broker. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems, AAMAS ’17*, pages 1–14. International Foundation for Autonomous Agents and Multiagent Systems, 2017.
 - [105] J. Hoogland and H. La Poutré. An effective broker for the power tac 2014. In *In S. Ceppi, E. David, C. Hajaj, V. Robu, & I. A. Vetsikas (Eds.). Agent-mediated Electronic Commerce. Designing Trading Strategies and Mechanisms for Electronic Markets*, page 66–80. Cham: Springer International Publishing, 2017.
 - [106] B. Liefers, J. Hoogland, and H. La Poutré. A successful broker agent for power tac. In *In S. Ceppi, E. David, V. Podobnik, V. Robu, O. Shehory, S. Stein, & I. A. Vetsikas (Eds.). Agent-*

mediated Electronic Commerce. Designing Trading Strategies and Mechanisms for Electronic Markets, page 99–113. Cham: Springer International Publishing, 2014.

- [107] P. P. Reddy and M. M. Veloso. Strategy learning for autonomous agents in smart grid markets. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Two, IJCAI'11*, pages 1446—1451. AAAI Press, 2011.
- [108] J. Serrano, , A. Y. R. González, and M. de Cote. Fixed-price tariff generation using reinforcement learning. *Fujita K. et al. (eds) Modern Approaches to Agent-based Complex Automated Negotiation. Studies in Computational Intelligence, Springer, Cham*, 674, 2017.
- [109] A. Sykulski, A. Chapman, E. Munoz de Cote, and N. Jennings. The winning strategy for the inaugural lemonade stand game tournament. In *Frontiers in Artificial Intelligence and Applications*, page 99–113, 2010.
- [110] S. Özdemir and R. Unland. Agentude17: A genetic algorithm to optimize the parameters of an electricity tariff in a smart grid environment. *Advances in Practical Applications of Agents, Multi-Agent Systems, and Complexity: The PAAMS Collection*, pages 224–236, 06 2018.
- [111] S. Jain, B. Narayanaswamy, and Y. Narahari. A multiarmed bandit incentive mechanism for crowdsourcing demand response in smart grids. In *AAAI Conference on Artificial Intelligence*, Canada, 2014.
- [112] J. Shweta and G. Sujit. A multiarmed bandit-based incentive mechanism for a subset selection of customers for demand response in smart grids. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 2046–2053, 2020.
- [113] H. Ma, V. Robu, N. L. Li, and D. C. Parkes. Incentivizing reliability in demand-side response. In *the proceedings of The 25th International Joint Conference on Artificial Intelligence (IJCAI'16)*, pages 352–358, 2016.
- [114] H. Ma, D. C. Parkes, and V. Robu. Generalizing demand response through reward bidding. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS'17*, pages 60–68, Brazil, 2017.
- [115] G. Methenitis, M. Kaisers, and H. La Poutré. Forecast-based mechanisms for demand response. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1600–1608, 2019.

- [116] Y. Li, Q. Hu, and N. Li. Learning and selecting the right customers for reliability: A multi-armed bandit approach. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 4869–4874, 2018.
- [117] J. V. Brakel. Robust peak detection algorithm using z-scores. 2014. <https://stackoverflow.com/questions/22583391/peak-signal-detection-in-realtime-timeseries-data> [Online; version: 2020-11-08].
- [118] M. Zeng, S. Leng, S. Maharjan, S. Gjessing, and J. He. An incentivized auction-based group-selling approach for demand response management in v2g systems. *IEEE Transactions on Industrial Informatics*, 11(6):1554–1563, 2015.
- [119] R. Zhou, Z. Li, and C. Wu. An online procurement auction for power demand response in storage-assisted smart grids. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 2641–2649, 2015.
- [120] A. Goudarzi, Y. Li, S. Fahad, and J. Xiang. A game theory-based interactive demand response for handling dynamic prices in security-constrained electricity markets. *Sustainable Cities and Society*, 72, 2021.
- [121] S. D. Ramchurn, P. Vytelingum, A. Rogers, and N. Jennings. Agent-based control for decentralised demand side management in the smart grid. In *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS ’11, pages 5–12, Richland, SC, 2011.
- [122] S. Jain, N. Balakrishnan, Y. Narahari, S. A. Hussain, and N. Y. Voo. Constrained tâtonnement for fast and incentive compatible distributed demand management in smart grids. In *Proceedings of the Fourth International Conference on Future Energy Systems*, pages 125–136. ACM, 2013.
- [123] Y.-Y. Hsu and C.-C. Su. Dispatch of direct load control using dynamic programming. *Power Systems, IEEE Transactions on*, 6(3):1056–1061, 1991.
- [124] H. Chao. Competitive electricity markets with consumer subscription service in a smart grid. *Journal of Regulatory Economics*, pages 1–26, 2012.
- [125] S. Park, Y. Jin, H. Song, and Y. Yoon. Designing a critical peak pricing scheme for the profit maximization objective considering price responsiveness of customers. *Energy*, 83:521–531, 2015.

- [126] X. Chen, Y. Nie, and N. Li. Online residential demand response via contextual multi-armed bandits. *arXiv preprint arXiv:2003.03627*, 2020.
- [127] A. Singh, P. M. Reddy, S. Jain, and S. Gujar. Designing bounded min-knapsack bandits algorithm for sustainable demand response. In *Pacific Rim International Conference on Artificial Intelligence*, pages 3–17. Springer, 2021.
- [128] Wikipedia contributors. Auction – Wikipedia, the free encyclopedia, 2023. [Online; accessed 5-December-2023].
- [129] R. B. Wilson. Chapter 8 Strategic Analysis of Auctions. *Handbook of Game Theory With Economic Applications*, 1:227–279, 1992.
- [130] N. Rashedi, M. A. Tajeddini, and H. Kebriaei. Markov Game Approach for Multi-agent Competitive Bidding Strategies in Electricity Market. *IET Generation, Transmission and Distribution*, 10(15):3756–3763, Nov. 2016.
- [131] A. Ghasemi, A. Shojaeighadikolaei, K. Jones, M. Hashemi, A. G. Bardas, and R. Ahmadi. A Multi-Agent Deep Reinforcement Learning Approach for a Distributed Energy Marketplace in Smart Grids. In *2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. IEEE, Nov. 2020.
- [132] Y. Du, F. Li, H. Zandi, and Y. Xue. Approximating Nash Equilibrium in Day-ahead Electricity Market Bidding with Multi-agent Deep Reinforcement Learning. *Journal of Modern Power Systems and Clean Energy*, 9(3):534–544, 2021.
- [133] B. Manvi and E. Subramanian. A Nash Equilibrium Solution for Periodic Double Auctions. In *2023 IEEE 62nd Conference on Decision and Control (CDC)*, 2023. Preprint available at Arxiv.
- [134] Y. Zhang, G. Qu, P. Xu, Y. Lin, Z. Chen, and A. Wierman. Global Convergence of Localized Policy Iteration in Networked Multi-Agent Reinforcement Learning. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 7(1):1–51, Feb. 2023.
- [135] Y. Yang and J. Wang. An Overview of Multi-Agent Reinforcement Learning from Game Theoretical Perspective. *CoRR*, abs/2011.00583, 2020.
- [136] D. Silver, A. Huang, and C. Maddison. Mastering the game of go with deep neural networks and tree search. volume 529, page 484–489. Nature, 2016.
- [137] R. D. McKelvey, A. M. McLennan, and T. L. Turocy. Gambit: Software Tools for Game Theory, Version 16.0.1. <http://www.gambit-project.org>, 2014. [Online; accessed 27-December-2021].

- [138] U.S. EIA. Peak-to-average electricity demand ratio rising in new england and many other u.s. regions. <https://www.eia.gov/todayinenergy/detail.php?id=15051>, 2014. [Online; accessed 19-January-2023].
- [139] International Energy Agency. The power to choose : Demand response in liberalised electricity markets, iea, paris, 2003.
- [140] Techopedia.com. Smart Grid. <https://www.techopedia.com/definition/692/smart-grid>, 2021. [Online; accessed 19-January-2023].