Large scale strategic decision making in multi-agent systems

Chen, Haipeng

2018

Chen, H. (2018). Large scale strategic decision making in multi-agent systems. Doctoral thesis, Nanyang Technological University, Singapore.

<http://hdl.handle.net/10356/74968>

<https://doi.org/10.32657/10356/74968>

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

**LARGE SCALE STRATEGIC DECISION
MAKING IN MULTI-AGENT SYSTEMS**

HAIPENG CHEN

INTERDISCIPLINARY GRADUATE SCHOOL

2018

LARGE SCALE STRATEGIC DECISION MAKING IN MULTI-AGENT SYSTEMS

HAIPENG CHEN

INTERDISCIPLINARY GRADUATE SCHOOL

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Doctor of Philosophy

2018

Abstract

With the rapid development over the recent years, artificial intelligence (AI) technologies (e.g., machine learning, data mining and multi-agent systems) are becoming more and more popular among large companies and governments. In both academy and industry, there exist vast research opportunities in the interdisciplinary areas of AI and traditional research (e.g., game theory, resource allocation, operations research and transportation). This thesis is targeted at engineering solutions methodologies across the range of possible multi-agent system requirements. The range coverage was achieved by considering specific significant representative problems such as the airlines industry, cloud computing and public transportation. More specifically, when making strategic decisions, the utility of one agent is usually affected by the decisions of multiple other agents. For example, the market share of an airline is not only affected by its own pricing strategy, but also by its competitors' pricing strategies. The large scale of the problem domain, the complex interactions among multiple agents, together with the uncertainties and dynamism from the environment, make this class of problems extremely challenging. To this end, this thesis investigates large scale strategic decision making in multi-agent systems, provides analysis over the problems and propose several solution algorithms to assist the agents in making intelligent decisions.

We start with investigating a relatively simple scenario where the strategies of other agents are fixed. This scenario is motivated by a practical problem where an airline wants to maximize its profit by optimally deciding its flight frequencies over its multiple routes, subject to certain budget constraints. In making optimal flight frequency strategies, there are two key issues. First, we need to solve a prediction problem, i.e., given a certain flight frequency over a route, what will be the profit of that route? Second, using the profit prediction as a sub-routine, we need to solve a resource allocation

problem, i.e., given a cost budget constraint, how to allocate different flight frequencies to different routes, so that the overall profit of an airline is maximized? To solve this problem, we propose the MAP (Maximizing Airline Profits) architecture, which consists of a novel Ensemble Forecasting (MAP-EF) approach and two frequency allocation algorithms, i.e., Bilevel Branch and Bound (MAP-BBB) and MAP-Greedy (MAP-G). Experimental results show that by using MAP, airlines can increase profits by a significant margin.

We then extend our framework to scenarios where there are interactions among the multiple agents, and the utility of one agent is dependent on the other agents' strategies. First, we extend our MAP architecture where all the airlines are strategic in deciding their flight frequencies. This problem is formulated as a competitive game, and we propose the ITER-Freq algorithm to compute the Nash equilibrium strategies of the competing airlines. We also provide extensive analysis (both empirically and theoretically) on the existence of Nash equilibrium as well as the performance of the algorithm. Second, we consider another practical, and yet more complex scenario in the cloud computing industry, where a private (secondary) cloud wants to enhance its service quality by 1) outsourcing workload to public (primary) clouds via vertical federation or 2) sharing resources with other secondary clouds through horizontal federation. We analyse the interrelated workload factoring and federation formation among secondary clouds, while providing scalable algorithms to assist them to optimally select partners and outsource workload. We use a game theoretic approach to model the federation formation of clouds as a coalition game with externalities. The results also show that compared with the two common practices, secondary clouds can decrease service delay penalty by around 11% with the proposed VHCF (Vertical and Horizontal Cloud Federation) framework.

Third, we consider sequential decision making in dynamic environments. We first focus on a dynamic tolling problem in a traffic network, which is motivated by deficiencies of Singapore's current static Electronic Toll Collection (ETC, also called Electronic Road Pricing in Singapore). We propose a novel dynamic ETC (DyETC) scheme which

adjusts tolls to traffic conditions in realtime. The DyETC problem is formulated as a Markov decision process (MDP), the solution of which is very challenging due to its high dimension state and action spaces. Due to the complexity of the formulated MDP, existing methods cannot be applied to our problem. Therefore, we develop a novel algorithm, PG- β (Policy Gradient method with Beta distribution based policy functions), which belongs to the family of reinforcement learning algorithms.

Finally, we study a scenario where self-interested agents make optimal sequential decisions in complex multiagent systems, and apply our designed algorithm on a representative example via a game scenario on the Minecraft platform. As an open problem for many years, various characteristics, such as complex interactions among agents, uncertainties, sequential decision making and limited learning trials all make it extremely challenging to find effective strategies. We make two key innovations to this class of problems. One key innovation is a generalized agent type hypothesis framework to identify the behavior model of the other agents, which is demonstrated to be robust to observation uncertainty. On top of that, a second key innovation is a novel Q-learning approach to learn effective policies against each type of the collaborating agents. Various ideas are proposed to adapt traditional Q-learning to handle complexities, including state-action abstraction to reduce problem scale, a warm start approach using human reasoning for addressing limited learning trials, and an active greedy strategy to balance exploitation-exploration. Applying our proposed solution algorithms to the Microsoft Malmo Collaborative AI Challenge (MCAC) in 2017, our agent HogRider won the first place among 81 teams from 26 countries.¹

¹Note that the author of this thesis is mainly responsible for designing the Q-learning framework and is involved in all the other aspects of designing the HogRider agent, credits should be given to all team members (especially Yanhai Xiong who has equal contributions as the author) who contributed to the work [1]. To conserve integrity of the paper and the corresponding idea, the author of this thesis has obtained agreement from the other team members for including the whole work in this thesis. Similarly, for the MAP work [2, 3] which is a teamwork of all the members, equal credits should be given to Noseong Park who is also entirely involved in the project.

Acknowledgements

It has been an amazing four years since I first came to Singapore. All the way through confusion and frustration, to enlightenment, joy and enthusiasm to both life and research, I am so grateful that I could meet the right people at the right time. Without these people, this work could never be done, and these four years could be totally different.

I would like to first thank my supervisor Soh Yeng Chai and my co-supervisor Bo An. Prof. Soh gave me valuable high level ideas of doing solid research, and freedom to explore various research directions. Whenever I was stuck, he was there to offer his advice and ensure that I was going in the right direction. Bo is the one that opens the door of research for me, so that I could see the beauty in it. Four years ago, I was a fresh graduate student who knew little about research. He taught me how to pick a research topic, formalize a problem, do literature review, search for a solution, how to conduct experiments, how to write a research paper, and how to present my ideas and communicate with other people. Whenever I had a problem with my research, he was always there to offer his help and advice. He also taught me his way of doing research through his own act and speech, which has deeply influenced my research philosophy. Bo is not only a teacher and advisor to me in research, but also a kind friend in my life. Every Chinese New Year, he would invite our AMI group to his home dinner. In these four years, AMI is family to me, and it will always be.

I would like to thank the directors of LILY center Miao Chunyan and Cyril Leung, and my mentor Ling Keck Voon. As senior professors, they showed me how to plan a research career, and how to stay passionate to research and life. I benefit a lot from your advice in pursuing my research career.

I would also like to express my gratitude to the other TAC members. I would like to thank my mentor, Prof. Ling Keck Voon for his continuous support throughout the four

years. I thank Prof. Guoqiang Hu for his advice on the high level structure of the thesis. I would express my gratitude to Prof. Zinovi Rabinovich for his insightful and very detailed comments. His comments helped a lot in improving the quality of the thesis.

I would like to thank my other co-authors for their valuable advice and guidance, including Noseong Park and V.S. Subrahmanian for the MAP project, Dusit Niyato for the VHCF work, Guni Sharon, Josiah P. Hanna and Peter Stone for the DyETC framework, Yanhai Xiong and Mengchen Zhao for the Malmo project. I would especially thank Noseong Park and V.S. Subrahmanian, from whom I developed passion and devotion to research.

Being in AMI, I am so fortunate to spend my four years with a group of passionate and talented young fellows, who are also my friends: Qingyu Guo, Yanhai Xiong, Mengchen Zhao, Zhen Wang, Jiarui Gan, Xinrun Wang, Zhenyi Wang, Martin Strobel, Youzhi Zhang, Ye Zhang, Wanyuan Wang, Jiuchan Jiang, Jinhua Song, Weixun Wang, Xu He, Lei Feng, Aye Phyu. I would always remember those days of catching all types of deadlines, research discussions, moments full of laugh and fun, and days of having lunch together. It has been a great joy with you guys. I would also like to thank all the people in LILY – a much larger and yet warm and kind family. Your simile everyday made my PhD life more colourful!

Finally, I would like to thank my parents and my brother for their ever-lasting love and support. I know how you are proud of me and how you always believe in me that I can do it. I am proud of you and will always believe in you. I thank my wife, Yanhai, who is also a kindly friend, a caring teacher and a patient listener to me. Thank you for walking with me throughout this process.

Contents

Abstract	i
Acknowledgements	iv
Contents	vi
List of Figures	x
List of Tables	xii
1 Introduction	1
1.1 Introduction	1
1.2 Targeted Problem Domains	2
1.2.1 Airline Profit Maximization with Flight Frequency Allocation	2
1.2.2 Vertical and Horizontal Cloud Federation	3
1.2.3 Electronic Toll Collection	5
1.2.4 Microsoft Malmo Collaborative AI Challenge	6
1.3 Large Scale Strategic Decision Making in Multiagent Systems	7
1.3.1 Data-Driven Decision Making in Large Scale Complex Systems	8
1.3.2 Strategic Decision Making in Multiagent Systems	8
1.3.3 Decision Making with Dynamism and Uncertainties	9
1.4 Main Contributions	10
1.5 Thesis Organization	12
2 Related Work	13
2.1 Large Scale Combinatorial Decision Making	13
2.1.1 Combinatorial Optimization	13
2.1.2 Data-Driven Decision Making	15
2.2 Game Theoretical Decision Making	16
2.2.1 Non-cooperative Games	16
2.2.2 Coalitional Games	17
2.2.3 Game Theoretic Decision Making in Cloud Computing	17
2.3 Sequential Decision Making	19
2.3.1 Approaches to Large Scale MDPs	19
2.3.2 Sequential Decision Making in Multiagent Systems	20

3	Data-Driven Decision Making	22
3.1	Existing Work on Airline Market Share and Demand Prediction	25
3.2	The MAP Dataset	26
3.3	Ensemble Prediction Model	27
3.4	Airline Profit Maximization via Frequency Allocation	32
3.4.1	Bilevel Branch and Bound (MAP-BBB)	35
3.4.2	Greedy Algorithm (MAP-G)	39
3.5	Experimental Results	41
3.5.1	Market Share & Total Demand Prediction	41
3.5.1.1	Experimental Environment	41
3.5.1.2	Experiment Results	41
3.5.2	Profit Maximization	43
3.5.2.1	Experimental Environment	43
3.5.2.2	Maximum Profit of Different Predictors	43
3.5.2.3	Comparing Different Optimization Algorithms	44
3.6	Chapter Summary	47
3.7	APPENDIX	48
4	Strategic Decision Making in Multiagent Systems	53
4.1	Frequency Allocation with Multiple Strategic and Competing Airlines .	53
4.1.1	Evaluation	55
4.2	Workload Factoring and Resource Sharing via Joint Vertical and Horizontal Cloud Federation Networks	58
4.2.1	Cloud Network System	58
4.2.1.1	Secondary Cloud and Horizontal Federation	59
4.2.1.2	Primary Cloud and Vertical Federation	59
4.2.1.3	Processing Time, Delay and Coalition Value Function	60
4.2.2	Coalition Formation and Workload Factoring of Secondary Clouds	61
4.2.2.1	Workload Factoring Game (WFG) for a Fixed Coalition Structure	62
4.2.2.2	Coalition Formation Game with Externalities (CFG-e)	63
4.2.3	Computing Least Pessimistic Core	68
4.2.3.1	Solving the Lower Level WFG	68
4.2.3.2	Solving the Upper Level CFG-e	74
4.2.4	Experimental Results	76
4.2.4.1	Obtaining Least Pessimistic Core: A Simple Scenario	76
4.2.4.2	Superadditivity and Externality	79
4.2.4.3	Coalition Formation Results under Different Cloud Network Conditions	80
4.2.4.4	Total Utility Comparison	82
4.2.4.5	Compare Different Algorithms	83
4.3	Chapter Summary	84
5	Sequential Decision Making with Dynamism and Uncertainties	86
5.1	Formulation of the DyETC Problem	88

5.1.1	Dynamic ETC System	88
5.1.2	An MDP Formulation	89
5.2	Solution Algorithm: PG- β	92
5.2.1	General Framework of PG- β	93
5.2.2	Policy Gradient for Time-Dependent MDP	94
5.2.3	A Novel Policy Function Based on the Beta PDF	95
5.2.4	State Abstraction	97
5.3	Experimental Evaluation	98
5.3.1	Evaluation on Synthetic Data	98
5.3.2	Evaluation on a Real-World Road Network of Singapore Central Region	102
5.4	A Cooking Assistance System for Patients with Alzheimer's disease	103
5.4.1	Formulation	105
5.4.1.1	Cooking System Formulation	105
5.4.1.2	MDP of Cooking Assistance Agent	106
5.4.1.3	State Transition Function	107
5.4.2	Analysis and Proposed Methods	108
5.5	Chapter Summary	110
6	Sequential Decision Making in Multi-Agent Systems	112
6.1	Challenge Description and Analysis	114
6.1.1	Settings & Rules	115
6.1.2	Analysis and Key Challenges	116
6.2	Solution Approach	117
6.2.1	Belief Update for Agent Types	117
6.2.2	Adapt Q-Learning to Various Challenges	119
6.2.3	HogRider	123
6.3	Experimental Evaluations	124
6.3.0.1	MCAC Results	125
6.3.0.2	Strategies for Selecting the Optimal Action	125
6.3.0.3	Belief Update	126
6.3.0.4	Warm Start	126
6.3.0.5	Different Exploration Methods	127
6.3.0.6	Compare HogRider with Human Players	128
6.4	Chapter Summary	128
6.4.1	Lessons Learned	128
6.4.2	Summary of Contributions	129
6.5	Appendix	129
7	Conclusion and Future Work	134
7.1	Conclusion	134
7.2	Future Work	136
7.2.1	Various Extensions	136

7.2.2	Applying Large Scale Decision Making Techniques to Health-care Problems	137
A	List of Publications	140
	Bibliography	141

List of Figures

1.1	A typical VHCF network. 4 secondary clouds form 2 horizontal federations. The horizontal federations can also rent resources from the primary cloud through a vertical federation.	5
1.2	ETC in Singapore	6
2.1	Quarterly revenue (dashed lines) and profit (solid lines) in 10 million dollars generated by MAP-EF. Revenue is defined as average ticket price multiplied by the number of passengers. Cost is per flight cost multiplied by frequency. Note that the profit-frequency curve (revenue minus cost) is very irregular even though the revenue-frequency curve is monotonically increasing. The shapes of the curves are quite different.	15
3.1	Structure of the MAP-EF prediction model. Rectangles mean data and ellipses algorithms. Each algorithm is marked with a processing sequence number. Solid lines mean inputs and dashed lines outputs. . . .	28
3.2	Quarterly revenue (dashed lines) and profit (solid lines) in 10 million dollars generated by MAP-EF. Revenue is defined as average ticket price multiplied by the number of passengers. Cost is per flight cost multiplied by frequency. Note that the profit-frequency curve (revenue minus cost) is very irregular even though the revenue-frequency curve is monotonically increasing. The shapes of the curves are quite different.	33
3.3	An example branch and bound process for MAP-BBB with three routes. Each blue bar of node represents a frequency range. Lower BB is used to calculate an upper bound profit of a node. c_l and c_r are two children of the parent node n_p	37
3.4	Best Profit-Cost Ratio among all routes in each iteration of the naive greedy Knapsack algorithm.	40
3.5	Profit maximization with different prediction methods (relative to MAP-EF) and 90% confidence interval is shown.	44
3.6	(a) Optimized profits and (b) Runtime for $K \leq 6$	45
3.7	(a) Optimized profits and (b) Runtime for $K \geq 5$,	46
3.8	Profits by varying price and frequency of other competitors	47
3.9	Scatter plot of market share predictions when $ \mathcal{A} \geq 4$. Each dot corresponds to a prediction of route.	52
4.1	Two steps to analyze the VHCF network.	62

4.2	Two cases of NE. x and y axes denote the request strategy of coalitions C_1 and C_2 , resp. The two lines L_1 and L_2 denote the <i>best responses curves</i> of coalitions C_1 and C_2 when the feasible region $\mathcal{A}$ is relaxed, while the dashed line segments are outside the feasible region. Following Theorem 4.15, they are replaced by the dotted line segments.	73
4.3	A CSG with 3 secondary clouds. The bottom node (level 1) is the grand coalition, the top node (level 3) has 3 singleton coalitions, and all nodes in level 2 have 2 coalitions.	75
4.4	Superadditivity & externality	80
4.5	Coalition formation results under different cloud network conditions	80
4.6	Total utility and total penalty of the secondary clouds	82
4.7	Runtime and least core values of different algorithms. A 90% confidence interval is shown.	83
5.1	Road network of Singapore Central Region	88
5.2	Event timeline of two subsequent time periods	90
5.3	Performance of different policy gradient methods	99
5.4	Traffic volume (in thousands) of existing tolling schemes under various traffic conditions	100
5.5	Total travel time (in thousands) of existing tolling schemes under various traffic conditions	100
5.6	Singapore Central Region road network	103
5.7	Performance of existing tolling schemes evaluated in Singapore Central Region	103
5.8	Statistics of Alzheimer's Disease	105
5.9	Steps of Cooking Spaghetti	105
6.1	Setting of MCAC. The left hand side of the figure is the first person view of the game, while the right hand side of the figure is the symbolic global view.	114
6.2	Two types of situations where a pig is caught	115
6.3	Probability distribution of the pig's positions after agent RED takes one step, where the grid in the center is the original position of the pig, and darker color means higher probability. In Figures 6.3a-6.3c, agent RED respectively waits for (a) no time (b) 1 second, and (c) 3 seconds before it takes a step. Note the pig does have a very short time (around 0.1 seconds) moving even if agent RED does not wait. This time is the response time of the Minecraft platform.	116
6.4	Flow of the human reasoning-derived decision tree	122
6.5	Evaluate human reasoning-aided initialization	126
6.6	Evaluate different exploration methods	127
6.7	Mean score curve for 3 strategies	130
6.8	Accumulated detection accuracy	131
6.9	Per episode results for Q-learning with warm start	131
6.10	Per episode results for comparison among 4 exploration methods	132

List of Tables

3.1	Summary of Notations	24
3.2	Variables used to make predictions by related works	27
3.3	Mean values of CC, R^2 and MAE/Max for market share predictions. $\uparrow$ indicates that larger values are preferred and vice versa. The best method is highlighted in yellow.	42
3.4	Variance of CC, R^2 and MAE/Max of Market Share Predictions.	42
3.5	CC, R^2 and MAE divided by true demand of Total Demand Predictions. Note that the variance of CC and R^2 cannot be defined because it predicts a scalar value for each route.	43
3.6	Profit maximization of MAP-G(BBB) with $q = 3$ (relative to the optimal profit) when $K = 4, 5$ or 6 . K is the total number of routes, q is the number of routes allowed in a group.	46
3.7	Variables to predict airline market share of R_1 by MAP-EF. "True Market Share" (highlighted) is the dependent variable to be predicted.	48
3.8	Variables to predict total demand of R_1 by MAP-EF. "True Total Demand" (highlighted) is the dependent variable to be predicted.	49
3.9	Mean values of CC, R^2 and MAE/Max for market share predictions. $\uparrow$ indicates that larger values are preferred and vice versa. The best method is highlighted in yellow.	50
3.10	Variance of CC, R^2 and MAE/Max of Market Share Predictions.	50
3.11	CC, R^2 and MAE divided by true demand of Total Demand Predictions. Note that the variance of CC and R^2 cannot be defined because it predicts a scalar value for each route.	50
4.1	The convergence behavior of Nash equilibrium frequency.	57
4.2	The profits obtained by airlines under equilibrium.	57
4.3	Summary of parameters mentioned in the experimental settings. We evaluate the influence of these parameter values on coalition formation results in Section 4.2.4.3.	77
4.4	NE, coalition value and least pessimistic core (the outcome highlighted in yellow).	78
4.5	Pessimistic value	78
6.1	Performance of different teams in MCAC. $\uparrow$ means the higher, the better, while $\downarrow$ means the opposite. The best performance is highlighted for all the tables.	125
6.2	Strategies for selecting the optimal action	125

6.3	Different methods for agent type belief update	126
6.4	Comparison with human players	128
6.5	Per episode results for different strategies	130
6.6	Per episode results for different belief update approaches	130
6.7	Per episode results for warm start	132
6.8	Per step results for warm start	132
6.9	Per episode results for exploration methods	133
6.10	Per step results for exploration methods	133
6.11	Per episode results V.S. human players	133

Chapter 1

Introduction

1.1 Introduction

Almost every organization needs to make certain planning decisions at some time points in managing its resources. For example, companies need to decide how to invest their limited budget into different projects, so that they could obtain the most profits. Police officers need to optimally allocate their patrolling forces to different spots, so that the city safety is ensured. Governments need to decide how to design the timing of the traffic lights, so that the traffic congestion is alleviated. In practice, those decisions and strategies are vital to the decision makers, and sometimes may even directly determine the survival of companies and the well-beings of the public.

Despite its significance, it is usually challenging to make optimal decisions due to 1) the lack of knowledge regarding the utility corresponding to a certain decision, and 2) the computational complexity brought by the huge problem scale. With its rapid development during the past decades, artificial intelligence is becoming more and more popular among large organizations in the decision making process, especially in cases of large scale problems where traditional approaches fail to find effective solutions. This problem belongs to the class of combinatorial optimization problems, which is known to be *NP-hard* and extremely challenging to compute optimal solutions. Moreover, in real-world scenarios, the utility of one organization (or agent) is not only dependent on its own decision, but also dependent on other agents' decisions. There are normally two

types of interactions among different agents, i.e., competition and cooperation. *Game theory* has been a well-established theoretical framework to model and analyse interactions among a group of agents [4–6]. In game theory, *Nash equilibrium* and the *core* concepts are used to define the solutions to competitive and cooperative games, respectively, while the situation where both cooperation and competition (called *coopetition*) exist has been paid less attention by the multiagent systems community. Although there are various studies regarding the coopetition phenomenon, they are usually empirical studies based on business case analysis [7–11], or studied from the perspective of physical dynamics of a coopetition network [12, 13], while to the best knowledge of the author, there is lack of theoretical analysis on the solution concept of a coopetition networks.

These two challenges are significantly amplified when it comes to sequential decision making, where there are uncertainties and dynamism in a system. There has been plenty of research work on single agent sequential decision making [14–19]. However, most existing work focuses on addressing small scale scenarios, and it remains a challenging problem for sequential decision making in complex and large scale scenarios. It is even more challenging when agents have to make sequential decisions in a multiagent system [20–27].

1.2 Targeted Problem Domains

In this thesis, we focus on developing novel techniques and solution algorithms to real world problems that are involved with strategic decision making in multiagent systems. Our study spans over a number of domains as described in the following.

1.2.1 Airline Profit Maximization with Flight Frequency Allocation

Since the deregulation of US airlines in 1978, there has been intense competition amongst airlines for market share and, eventually, profitability. For an airline, one critical aspect is to decide the number of flights (i.e., flight frequency) at each of its

operating routes. Intuitively, with larger flight frequencies an airline will have a larger market share and total demand (i.e., more sales of tickets) and thus higher revenue. Meanwhile, the operation cost grows with respect to flight frequencies. As a result, optimal flight frequency decisions should balance the trade-off between sales of tickets and operation cost budget, which depend on three critical aspects. First, given a flight frequency on a route, it is important to precisely predict the market share and total demand. While there is considerable work on predicting market share and demand along airline routes, the prediction of existing models is not precise enough and, to the best of our knowledge, there is no use of data-mining based forecasting techniques to improve airline profitability. Second, it is a combinatorial optimization problem to find the optimal allocation of cost budgets in order to maximize an airline's profit, which is known to be NP-hard. Last, while deciding frequency strategies of one airline, it is important to take into consideration the frequency strategies of other competing airlines.

1.2.2 Vertical and Horizontal Cloud Federation

Cloud computing has emerged as a popular service model where computing resources can be accessed by users over the Internet on a usage basis, avoiding upfront investment on the physical infrastructures. It has been shown that individual private clouds (*secondary clouds*) can form *horizontal* cloud federations [28] and scale up their services by pooling their computing resources. Furthermore, the horizontally federated clouds can vertically outsource part of their requests to a public cloud (*primary cloud*) and pay the usage fee, which is called workload factoring/outsourcing. For example, in Fig. 1.1, if secondary cloud A receives 80 requests from its end users and has only 60 computing resource units (e.g., CPU, memory, and bandwidth), while secondary cloud B receives 20 requests and has 30 resource units, then B can share its redundant resources to process the excessive requests from A and receive a share of payoff from A, while for A, this payoff share is usually lower than the cost of outsourcing to primary clouds.

Meanwhile, the total number of requests is still larger than the total number of resources. To finish the requests on time and avoid the penalty of violating *service level agreements (SLAs)*, the formed horizontal federation can further outsource their requests

to the primary cloud via a vertical federation. Different from the resource sharing in a horizontal federation, the resources of the primary clouds are on a renting basis and are often costly. Moreover, the resources in a horizontal federation are dedicated to processing internal requests, while in a vertical federation, the resources of a primary cloud are shared by different external requests. Therefore, the performance (i.e., processing time) of the requests in a primary cloud depends on the number of external requests outsourced by other entities [29]. While horizontal federations may still experience resource shortage or redundancy, vertical federation is a perfect complement. Due to the merits, such a joint *vertical and horizontal cloud federation* (VHCF) network is evaluated both theoretically [30] and practically (e.g., Aristotle federated cloud [31] and EUBrazil Cloud Connect [32]).

Although the technical aspects of both the vertical and horizontal federations have been investigated [30, 33, 34], the economic aspects of forming such a joint cloud federation have never been studied, while these individual private clouds, which can be treated as *intelligent agents* in a network, are usually self-interested. To this end, three intrinsic questions remain to be answered: 1) *Federation Formation*. If cooperation incurs a cost, what is the stable cooperative pattern? Is the grand coalition¹ - a coalition of all the clouds - always stable? How much benefit will each coalition obtain, and how to divide the benefit among the coalition members so that nobody wants to deviate? 2) *Workload factoring*. Since a coalition's task performance in a primary cloud depends on the strategies of the competing coalitions, what is the optimal workload factoring strategy of each coalition? 3) *Efficiency*. Is it beneficial for a group of secondary clouds to employ such a VHCF network?

Due to the interdependency of the competitive workload factoring among different coalitions and the cooperative resource and payoff sharing within a coalition, it is not easy to answer the aforementioned questions. Recently, several game theoretic approaches have been proposed to study the economics of cloud computing [29, 35–45] and other forms of networks [46–52]. Unfortunately, these approaches ignored either

¹We use “horizontal federation” and “coalition” interchangeably, to denote a group of cooperating secondary clouds. Note that we do not treat “vertical federation” as a coalition, since the “sharing” of resources via vertical federation is on a renting basis.

Figure 1.1: A typical VHCF network. 4 secondary clouds form 2 horizontal federations. The horizontal federations can also rent resources from the primary cloud through a vertical federation.

cooperation or competition among the players, and thus their methods cannot be applied.

1.2.3 Electronic Toll Collection

Nowadays, governments face a worsening problem of traffic congestion in urban areas. To alleviate road congestion, a number of approaches have been proposed, among which ETC has been reported to be effective in many countries and areas (e.g., Singapore [53], Norway [54]). Since 1998, an ETC system has been deployed in Singapore to alleviate its traffic congestion, especially in the central region (Figure 1.2a). In an ETC system, vehicles are charged when they pass through ETC gantries (Figure 1.2b) and use priced roads during peak hours. Typically, ETC rates vary from different roads and time periods, which encourages vehicles to change their travel routes in order to alleviate traffic congestion. As shown in Figure 1.2c, while an ETC gantry charges different ETC rates at different time periods, the rates are fixed during certain time periods from Monday to Friday. This framework, which is demonstrated to have significantly improved Singapore's traffic, still has two major shortcomings. *First, the current ETC system is determined based on the historical traffic flows, which fails to adjust to the uncertainty of traffic demand.* In reality, the real-time traffic demand could fluctuate. For example, during morning rush hour, we may know the average traffic demand based on history, but the exact demand cannot be precisely predicted. As a result, if we set tolls

merely according to average traffic demand but the real demand is not as severe as expected, few vehicles will go through this road even if the congestion is not severe since they are scared off by the high tolls. Under such cases, the current ETC system will have less positive or even negative effect on the traffic condition. *Second, current ETC rating systems fail to precisely respond to the extent to which the traffic is congested.* As we can see from Figure 1.2c, the interval of the current tolling scheme is 0.5. For example, the tolls are either 0 or 0.5, while the optimal toll might be somewhere between these two rates. In such cases, the current tolling scheme can hardly reveal and react to the accurate congestion level. Motivated by this, we extend our one-shot decision making framework, and consider *sequential decision making* in a dynamic environment.

Figure 1.2: ETC in Singapore

1.2.4 Microsoft Malmo Collaborative AI Challenge

In complex multiagent systems, sequential decision making has long been a significant challenge with many characteristics. One prominent characteristic is the *interactions* among the agents. In many practical scenarios (e.g., hunting), agents need to *cooperate* with each other to achieve a common goal with high rewards (e.g., hunting stags), while each individual agent is *self-interested* and might deviate from the cooperation for less risky goals with low rewards (e.g., hunting rabbits).

Another critical characteristic comes from various *uncertainties*. One type of uncertainty arises from the lack of accurate knowledge of environment and other agents where the uncertain information can be modelled probabilistically. A more challenging

type of uncertainty lies in the environment-related factors which we do not know how to model.

These two challenges are significantly amplified when it comes to *sequential decision making*, where we need to look at not only short term rewards but also rewards in the long run. Therefore, one has to consider subsequent effect of the current action, especially in a dynamic environment. Another crucial characteristic is the *limited learning trials*. On the Minecraft platform, it usually takes several seconds to complete one episode of game. Therefore, it is extremely time consuming to learn an effective policy.

The Microsoft Malmo Collaborative AI Challenge (MCAC), which is designed to encourage research relating to various problems in Collaborative AI, builds on a Minecraft mini-game [55] called “Pig Chase”. Pig Chase is played on a 9×9 grid where agents can either work together to catch the pig and achieve high scores, or give up cooperation and achieve low scores. After playing certain episodes (e.g., 100) of games, the agent who achieves the highest average scores wins the challenge. Despite its simple rules, this game has all the above stated key characteristics. Though there are numerous papers studying sequential decision making in complex multiagent systems, they only address a subset of these characteristics. We hope to shed some light on solving this class of problems by presenting HogRider, a champion agent which won 2017 Microsoft Malmo Collaborative AI Challenge [56].

1.3 Large Scale Strategic Decision Making in Multiagent Systems

Large scale decision making in multiagent systems has long been a challenging problem over the past decades, with a number of open problems and common issues that remain imperfectly solved. Meanwhile, there have been numerous attempts to solving this class of problems, which leave a solid theoretical foundation as well as a range of concepts and solution techniques to this research area. In this thesis, we aim to investigate several important aspects that are related to decision making problems in multiagent systems, and develop novel frameworks and solution algorithms to these problems.

1.3.1 Data-Driven Decision Making in Large Scale Complex Systems

Large scale decision making problems are ubiquitous in real world, involving all kinds of areas such as transportation regulation, assembly line control, budget allocation, resource allocation, electronic commerce, . . . Despite its broad existence, the difficulty of making decisions in large scale complex systems has long been studied and recognized. Those difficulties include:

- *The large scale of the complex system.* Take the airline profit maximization problem as an example, for an airline that operates on dozens of routes, deciding the optimal flight frequency for each of the routes requires solving a *knapsack problem*, which belongs to the family of combinatorial optimization problems, and is known to be *NP-hard*.
- *A lack of knowledge of the accurate utility of a certain decision.* This is quite common in reality. For example, in the airline profit maximization problem, it is not straightforward to know that, given a specific frequency strategy on a certain route, what will be the profit of that route.

1.3.2 Strategic Decision Making in Multiagent Systems

Another type of difficulty comes from the interactions among other agents who are in the same system. Typical scenarios exist in areas such as commodity pricing, strategic level of national defense, resource sharing, . . . There are mainly two types of interactions amongst a group of agents:

- *Competition.* In the airline profit maximization problem, airlines need to compete with each other for market share. For an airline, given a same flight frequency strategy on a specific route, its market share (and thus profit) is also dependent on the frequency strategies of the other competing airlines who are also operating on that route. Intuitively, if other airlines invest more flight frequencies on that route, the market share of the target airline will decrease (and vice versa).

- *Cooperation.* When in cooperation, the utility of an agent usually depends on 1) who that agent is cooperating with and 2) what the other agent offers and requires in the cooperation.

The problem gets even more complex when there are both competition and cooperation among the agents. In the vertical and horizontal cloud federation (VHCF) example, the utility of a secondary cloud provider depends on both 1) which horizontal federation it belongs to (i.e., it depends on the other cloud providers in the same cloud federation), and 2) for the cloud providers that are not in the same cloud federation, what workload factoring (i.e., vertical federation) strategies they will use.

1.3.3 Decision Making with Dynamism and Uncertainties

The difficulties are significantly amplified when it comes to sequential decision making, with dynamism and uncertainties. Normally, there are two types of uncertainties:

- *Uncertainties of the environment.* One type of uncertainty arises from the lack of accurate knowledge of environment where the uncertain information can be modelled probabilistically. A more challenging type of uncertainty lies in the environment-related factors which we do not know how to model.
- *Uncertainties of the other agents.* In many cases, there is no prior knowledge about the characteristics of the other competing/cooperating agents. In the Microsoft Malmo Collaborative AI Challenge, the type of the other agent is not known in advance. As a result, it is necessary to find an agent-characteristic hypothesis framework to efficiently and accurately detect the characteristics of the other agents.

Similarly, there are two sources of dynamism in a multiagent system:

- *Dynamism of the environment.* After taking an action (according to the decision made), there is interaction between the agent and the environment, which leads to dynamic changes (usually called state transition) to the environment. These

dynamic changes could either be deterministic or stochastic, where are usually modelled as a Markov Decision Process (MDP).

- *Dynamism of the agents.* In reality, agents may have evolving characteristics such as objectives, habits, preferences, which further adds to the difficulty of correctly detecting the type of other agents. This type of dynamism of is out of scope of this thesis.

1.4 Main Contributions

This thesis makes several important contributions to the state of the art in the area of large scale strategic decision making in multiagent systems, from both a theoretical and a practical point of view. We summarize the major contributions of this thesis as follows:

- We propose the MAP (Maximizing Airline Profits) architecture designed to help airlines and make two key contributions in *airline market share and route demand prediction* and *prediction-based airline profit optimization*. Compared with past methods to forecast market share and demand along airline routes, we introduce a novel Ensemble Forecasting (MAP-EF) approach considering two new classes of features: (i) features derived from clusters of similar routes, and (ii) features based on equilibrium pricing. Using the results of MAP-EF, we develop MAP-Bilevel Branch and Bound (MAP-BBB) and MAP-Greedy (MAP-G) algorithms to optimally allocate flight frequencies over multiple routes, to maximize an airline's profit. Experimental results show that airlines can increase profits by a significant margin.
- We extend the MAP framework to consider competitions among agents, and develop ITER-Freq algorithm to compute Nash equilibrium strategies for the competing airlines. We also consider a more complicated multiagent system scenario in the cloud computing area, where both competition and cooperation exist. We

analyze the interrelated workload factoring and federation formation among secondary clouds, while providing scalable algorithms to assist them to optimally select partners and outsource workload. We use a game theoretic approach to model the federation formation of clouds as a coalition game with externalities. We adopt a pessimistic core to characterize the cooperation stability and formulate its computation as a bilevel optimization problem. The properties of the problem are explored and efficient algorithms are developed to solve it.

- We further study large scale sequential decision making in a dynamic environment. Applying our framework on Electronic Toll Collection, we propose a novel dynamic ETC (DyETC) scheme which adjusts tolls to traffic conditions in real-time. The DyETC problem is formulated as a Markov decision process (MDP), the solution of which is very challenging due to its 1) multi-dimensional state space, 2) multi-dimensional, continuous and bounded action space, and 3) time-dependent state and action values. Due to the complexity of the formulated MDP, existing methods cannot be applied to our problem. Therefore, we develop a novel algorithm, PG- β , which makes three improvements to traditional policy gradient method by proposing 1) time-dependent value and policy functions, 2) Beta distribution policy function and 3) state abstraction.
- Finally, we consider sequential decision making in a multiagent system, and apply our designed agent HogRider to the Microsoft Malmo Collaborative AI Challenge 2017. One key innovation of HogRider is a generalized agent type hypothesis framework to identify the behavior model of the other agents, which is demonstrated to be robust to observation uncertainty. On top of that, a second key innovation is a novel Q-learning approach to learn effective policies against each type of the collaborating agents. Various ideas are proposed to adapt traditional Q-learning to handle complexities in the challenge, including state-action abstraction to reduce problem scale, a warm start approach using human reasoning for addressing limited learning trials, and an active greedy strategy to balance exploitation-exploration.

1.5 Thesis Organization

The remainder of this thesis is organized as follows. In Chapter 2, we describe existing work that is related to large scale decision making in multiagent systems. In Chapter 3, we study a relative simple scenario of large scale data-driven decision making in a single agent environment, and provide efficient solution algorithms to the formulated problem. We then extend the single agent scenario to multiagent scenarios in Chapter 4. We propose both theoretical analysis to the problem and novel approaches to compute the optimal solution. In Chapter 5, we further consider large scale, sequential decision making problems in an uncertain and dynamic environment, followed by Chapter 6, where we study large scale, sequential decision making in multiagent systems. We provide novel and efficient solution algorithms to the problems formulated in both of these two chapters. Finally, we conclude the key aspects of this thesis in Chapter 7 and discuss future directions.

Chapter 2

Related Work

In this chapter, we review existing research that is relevant to this thesis.

2.1 Large Scale Combinatorial Decision Making

There have been extensive research efforts on strategic decision making in large scale complex systems, e.g., in water systems [57], power systems [58], cloud networks [59], and manufacturing [60]. In this thesis, we are particularly interested in a specific class of large scale decision making problems called *combinatorial optimization*.

2.1.1 Combinatorial Optimization

Many real-world problems could be classified as combinatorial optimization problems, such as *set partitioning problem*, *travelling salesman*, *task scheduling*, and *knapsack problems*. Typical approaches to solve the class of combinatorial optimization problems include *greedy algorithms* [61, 62], *approximation algorithms* with performance guarantees [63], *local search* [64], *linear programming* [65], *dynamic programming* [66], and *randomized algorithms* [67]. We refer to [68] for a detailed description of the class of combinatorial optimization problems. In this thesis, we are particularly interested in the knapsack problems (KPs) and resource allocation problems (RAPs) which are

closely related to the airline frequency allocation problems described in the motivating examples.

KPs and RAPs. Airline profit maximization through optimal frequency allocation over multiple routes, given a certain budget constraint, belongs to the family of KPs and RAPs, which have been extensively studied ever since the work of Dantzig [69] in 1957. While all variants of KPs are known to be *NP-hard*, MAP, which has bounded integer variables, and a non-concave (arbitrary) objective function, is notoriously hard [70].

There are many existing algorithms for bounded knapsack problems (BKPs). Using a greedy algorithm, Keller et al. [71] design a Polynomial Time Approximate Scheme (PTAS) for BKPs with a runtime of $O((nl)^l)$, where $l = \min\{1/\lceil \varepsilon \rceil - 2, n\}$. However, their greedy algorithm is designed for BKPs with a linear objective function, and thus is not applicable to our problem with non-linear objective function. Pisinger [72] proposes a dynamic programming algorithm for the BKPs with a theoretically bounded complexity of $O(nc^2)$, where c is the upper limit of quantities in each item category. It is improved by Pferschy [73] who solves BKPs in $O(nc)$ time. However, both these two algorithms need to order items according to decreasing efficiencies, which also require linear objective functions and thus cannot solve our problem. A number of branch and bound algorithms have also been proposed [72, 74], most of which find upper bounds of objectives using linear relaxation. However, in our problem, the objective function is not as simple as other functions that can be solved with linear programming. Apart from these algorithms specifically designed for BKPs, another thread of works consider transforming BKPs into the much simpler 0/1 KPs [75]. The main idea is, for each item in the original BKP, to construct $\log(c)$ “virtual” items in the new 0/1 KP. However, this transformation also requires linear objective functions. For non-linear objective function, it constructs c virtual items for each original item, which yields no computational gains. Apart from these algorithms specifically designed for BKPs, another thread of works consider transforming BKPs into the much simpler 0/1 KPs [75].

Unfortunately, none of these works for the airline frequency allocation problem. This is because the profit-frequency function in our problem is neither concave nor convex (thus not linear), varies from one route to another, and cannot be described by a simple math formula but by a series of clustering, regression, and other types of functions.

Figure 2.1: Quarterly revenue (dashed lines) and profit (solid lines) in 10 million dollars generated by MAP-EF. Revenue is defined as average ticket price multiplied by the number of passengers. Cost is per flight cost multiplied by frequency. Note that the profit-frequency curve (revenue minus cost) is very irregular even though the revenue-frequency curve is monotonically increasing. The shapes of the curves are quite different.

As shown in Figure 3.2 where the x-axis is the frequency and the y-axis is the profit (or revenue), the two graphs are for two routes (SFO-LAX route on the left and JFK-LAX route on the right) and 4 major US airlines. We see that the shapes of the curve on the left (SFO-LAX) is quite different from another on the right (JFK-LAX). In fact, across the 700 routes we studied, there was huge variation in the actual profit/revenue curves. Another alternative is to formulate the MAP problem as a multi-objective KP [76], with the revenue and cost as multiple objectives to optimize. However, given the nature of the problem (that revenue and cost are equally important), the Pareto-optimality concept for the multi-objective KP is not equivalent to the optimality concept in this problem. Thus, any method specially devised for a certain form of objective functions (e.g., linear, concave or convex) cannot be applied.

2.1.2 Data-Driven Decision Making

Decision making is usually the goal of data-mining approaches. Nowadays, more and more organizations make decisions with the assistance of big data. Data-driven approaches are broadly employed in various domains such as supply chain management [77], education [78], optimal bidding [79, 80], smart pacing [81] and optimal

recommendation systems [82]. In the first step of these works, an objective function is learned using regression analysis techniques. Based on the prediction results, an optimization problem is solved, either analytically [79, 82] or via optimization algorithms [80, 81]. The airline frequency allocation problem resembles this class of problems. However, this problem is significantly distinct from the above mentioned problems, in the sense that 1) the objective function in the airline frequency allocation problem is a very complex linear algebra form, rather than a simple linear/convex regression form, and 2) the optimization problem itself is a challenging combinatorial optimization problem. These two properties make the existing approaches intractable to the airline frequency allocation problem.

2.2 Game Theoretical Decision Making

In this section, we first introduce the basics of non-cooperative and cooperative games, and then we describe related works on game theoretic approaches to the area of cloud computing.

2.2.1 Non-cooperative Games

In game theory, a *non-cooperative game* consists of a set n of *players*, $\{1, \dots, n\}$. Each player i has a set of possible *strategies*, S_i . To play the game, each player i selects a strategy $s_i \in S_i$. $s = (s_1, \dots, s_n)$ denotes the vector of strategies selected by the players and $S = \times S_i$ denotes the set of all possible ways in which players can pick strategies, i.e., *strategy space*. The strategies $s \in S$ of the players will determine the utility for each of the players. Denote the utility of player i as $u_i(s)$, in a non-cooperative game. In non-cooperative game, *Nash equilibrium* is a natural solution concept, where *pure strategy Nash equilibrium* represents a vector of strategies for the players, where no player i could increase his utility from unilaterally changing his own strategy s_i to any alternate strategy s'_i , i.e.,

$$u_i(s_i, s_{-i}) \geq u_i(s'_i, s_{-i}), \quad (2.1)$$

where s_{-i} denotes the strategies selected by all the other players except player i .

There have been extensive efforts on the research of non-cooperative games, including the existence of Nash equilibrium [4], uniqueness and stability of Nash equilibrium [5], and algorithmic aspects of game theory [6]. There are variants of solution concepts for the non-cooperative games such as Stackelberg equilibrium [83], Cournot equilibrium and Bertrand equilibrium [84], we omit the introduction of these concepts here since they are out of scope of this thesis.

2.2.2 Coalitional Games

In this thesis, we focus on coalitional game, which belongs to a larger family of cooperative games. In coalition games, consider a set N of players. For each subset $A \subseteq N$ is associated with a value $v(A)$. The problem is to divide the value with the set N of players in a "fair" or "stable" way. Denote the division of value for each player i as x_i , an division outcome is denoted as the vector $x = (x_1, \dots, x_n)$. For *stability*, the "core" concept is defined [85]:

Definition 2.1. A division outcome x is **in the core** if $\sum_{i \in A} x_i \geq v(A)$ for all sets $A \subseteq N$.

For *fairness*, Shapley value [86] is usually utilized to denote the division outcome where the value divided by each player is proportional to each player's incremental contribution to the coalition's utility.

2.2.3 Game Theoretic Decision Making in Cloud Computing

Of special interest in this thesis is the applications of game theoretical analysis in the domain of cloud computing. Works related to economics of cloud computing can be categorized into topics including resource allocation/task scheduling [35, 36], pricing [37, 38] and workload factoring/outsourcing [29, 39, 40]. The VHCF problem

belongs to the last category. There have been a few works on optimizing cloud outsourcing. In [39], the authors aim at maximizing utilization of internal data center, minimizing cost of running outsourced tasks, while maintaining service quality. In [40], workload factoring is considered in combination with the task scheduling problem within a private cloud. These works neglect the interrelated outsourcing behaviors among different secondary clouds. In [29] the authors study the workload factoring among users of the same cloud provider. A non-cooperative game is formulated where cloud users make strategic decisions on workload factoring. However, they do not consider cooperation among the cloud providers.

Competition and cooperation among different cloud providers have been independently studied in other economic aspects of cloud computing. In [41], price competition is considered in the setting of multiple cloud providers. Instead of studying the competition among cloud providers, in [42], the competition among the cloud users is investigated. These two works focus on competition among cloud providers or users, but they ignore their potential cooperation. Cooperation among clouds is studied by applying cooperative game theory. In [43] the resource and revenue sharing problem within a horizontal cloud federation is formulated as a stochastic program. In [44], a coalition game based resource allocation and revenue sharing scheme is proposed. The stable coalition structure is obtained with the merge-split algorithm. The authors in [45] introduce an MILP-based formulation of the energy-aware resource allocation within a coalition, which aims at minimizing energy cost of a coalition. However, these works only study the cooperative behavior among cloud providers.

Game theory has been extensively employed to analyze cooperation and competition in other forms of networks. In [46], the authors study the coalition formation game among small cell base stations for joint beamforming in cell networks. An overlapping coalition formation game model is proposed in [47] to study the cooperation among secondary users in cognitive radio networks, while in another work [48], the coalition formation among primary users is studied. In [49], the authors model the cooperation of the small-scale electricity suppliers and the end users as a coalition game. In [50], the spectrum sharing among wireless users in a wireless network is modelled as a non-cooperative game. The authors in [51] study a non-cooperative resource allocation game

among the device-to-device users in cell networks. In [52] a non-cooperative spectrum load balancing game among the secondary users is proposed in a cognitive radio network. These works mainly focus on resource contention among different entities.

Despite the various studies regarding the coopetition phenomenon, they are usually empirical studies based on business case analysis [7–11], or studies from the perspective of physical dynamics of a coopetition network [12, 13]. To the best of our knowledge, there is lack of theoretical analysis on the solution concept of a coopetition networks. We aim at filling the gap.

2.3 Sequential Decision Making

In this section, we first introduce the line of research on solving large scale MDPs, which is of close relation to the motivated ETC problem. We then discuss related works of sequential decision making in multiagent systems.

2.3.1 Approaches to Large Scale MDPs

Solution algorithms to MDPs have long been explored by both the academic and industrial communities over the past few decades. Dynamic programming [14, 87] and linear programming [88, 89] are the most common approaches to solving MDPs, but are intractable for MDPs with large scale state and action spaces due to the curse of dimensionality.

Recently, several studies have been proposed to solve large scale MDPs using reinforcement learning. Traditional reinforcement learning algorithms based on tabular representations of the value and policy functions (e.g., *tabular Q-learning* [15], *prioritized sweeping* [16], Monte Carlo Tree Search (MCTS) [17] and UCT (UCB applied to trees) [18]) cannot be applied to our problem due to the large scale state and action spaces. Value-based methods with function approximation [90–93], which represent the state-action values (often referred to as “Q-values”) with function approximators are

also inefficient due to the complexity in selecting the optimal action in a continuous action space. While *policy gradient methods* [19, 94–96] work well in solving large scale MDPs with continuous action space, current policy gradient approaches usually focus on MDPs with unbounded action space. To handle bounded action spaces, Hausknecht and Stone [97] propose three approaches (i.e., zeroing, squashing and inverting) on the gradients which force the parameters to preserve in their intended ranges. These manually enforced constraints may deteriorate the solution’s optimality.

2.3.2 Sequential Decision Making in Multiagent Systems

One line of related research is teamwork within a multiagent system where agents cooperate through communication [20, 21] or pre-coordination [22]. These methods assume fully cooperative agents and do not consider uncertainty about cooperating agents’ types. Recently, research efforts have been put on ad hoc interactions among agents, where agents need to interact without prior knowledge of the other agents [98]. They study both cooperative [99, 100] and competitive [100, 101] scenarios, where agent type hypothesis and opponent modelling are used to handle the uncertainty. However, they do not consider the uncertainty in agent action observation and limited learning trials.

Multi-agent reinforcement learning (MARL) has also been applied to study interactions among a set of agents [23–25]. These approaches, from dealing with small to large scale problems, usually focus on obtaining Nash equilibrium via self-play, or learning optimal strategies against stationary opponents.

Another line of research studies the sequential multiagent planning problem via game theoretic approaches [26, 27]. Brafman et al. [102] apply game theoretic solutions to coalition planning games and auction planning games. Jordán and Onaindia [103] propose a two-game proposal for solving congestion games among a group of autonomous vehicles [104].

Some other papers [105, 106] apply a stochastic game framework to multiagent planning problems. Despite the theoretical merits, pure game theoretic approaches usually suffer from computational issues that arise from large-scale state and strategy

spaces. Other classical approaches to sequential planning in multiagent systems include decentralized partially observable Markov decision process and its variants [107, 108]. These approaches also suffer from high computational complexity.

Chapter 3

Data-Driven Decision Making

The first problem that we address in this thesis is large scale decision making with the help of data-driven prediction. One example in the real world that motivates this work is the frequency-based airline profit maximization problem. Since the deregulation of US airlines in 1978, there has been intense competition amongst airlines for market share and, eventually, profitability. While there is considerable work on predicting market share and demand along airline routes, the prediction of existing models is not precise enough and, to the best of our knowledge, there is no use of data-mining based forecasting techniques to improve airline profitability. In this work, we formally define the MAP (Maximizing Airline Profits) problem by allowing airlines to decide the flight frequencies on their flying routes in order to maximize their profits, subject to cost constraints. For instance, if there is a set $\mathcal{R}$ of routes in the world, and the airline can fly up to n routes where $n \leq |\mathcal{R}|$, then our MAP problem would capture both route selection (which routes to fly) and frequency¹.

The first key contribution of this chapter is a precise route-specific prediction of both total demand of an origin-destination market and an airline's market share. Existing related works only use simple regression methods and focus on a limited number of variables. Our studies of three major route market share [109–111] and demand prediction [112–114] methods show that the Pearson Correlation Coefficients between the

¹Because the airline industry is very complex, it is difficult to model all other sources of profit that might be available (e.g., in-flight sales, baggage fees, etc.) to the airline, especially as such data is not freely available.

predicted values and the actual values are a max of 0.82 (for one market share method) and 0.77 (for one demand model). Other existing models predict much lower numbers. Although one of these regression methods [109] outputs satisfactory prediction results in our experiments, there is still a gap between the prediction and the real value. The relatively “small” prediction gap can lead to a huge revenue loss if airlines make decisions based on the prediction. Worse still, due to the inflexibility of existing models, the prediction gap can be large for some routes. We propose a new ensemble-based prediction method for forecasting total demand and market share, which uses an extensive number of features and several state-of-the-art clustering and regression algorithms that have never been adopted for MAP. The new ensemble method builds upon existing models, but also collects several new features, together with novel clustering and game theoretic methods. We are the first to propose a prediction method, for a broad set of routes (around 700 - past works stopped at 200), which considers the predictions of both total demand (demand generation) of a route and market share (demand allocation) of each airline operating in that route (13 airlines in total).

The second key contribution is two novel algorithms for solving MAP, which becomes computationally intractable with brute-force search when the number of routes is large. This is because: 1) the solution space is exponential w.r.t. the number of routes, 2) the profit-frequency function, which is generated by our proposed prediction method, is neither convex nor concave (and thus not linear). As discussed in Chapter 2, MAP belongs to the hardest subclass of *Knapsack Problems* (KPs) and *Resource Allocation Problems* (RAPs), and despite a vast amount of existing works [70, 72, 73, 115] for both KPs and RAPs, all previous algorithms fail to solve it efficiently. Based on predictions made by MAP-EF MAP-Ensemble Forecasting, we come up with two optimization algorithms to solve the raised profit maximization problem. We present an exact algorithm to solve MAP based on a novel Bilevel-Branch and Bound approach (MAP-BBB) that computes the true optimal solution, as well as a Greedy algorithm (MAP-G) that more quickly computes suboptimal solutions.

Third, we conduct extensive experimental evaluations to compare MAP’s prediction results with those of past works. We show that past prediction models are significantly

“beaten” by MAP, which increases these predictive accuracies to over 0.95, while significantly reducing the variance in our prediction error compared with past works. We also compare both the optimality and scalability of our proposed profit maximization algorithms with several benchmarks. The result shows that our predictions are far superior to past efforts. Moreover, by using MAP, an airline can averagely increase its profit by at least 55% under mild conditions.¹

$\mathcal{A}$	a set of airlines
A_i	i^{th} airline of $\mathcal{A}$
$\mathcal{R}$	a set of routes
R_i	i^{th} route of $\mathcal{R}$
Symbols related to predict market share of route R	
X_{ik}	the value of the k^{th} variable, e.g., price, of airline A_i
η_k	the weight of X_{ik}
V_i	airline A_i 's customer utility in R
m_i	market share of airline A_i in R
Symbols related to predict demand of route R	
Y_k	the value of the k^{th} variable, e.g., population
λ_k	the exponent of Y_k
d	total demand (the number of passengers) of route R
Symbols related to calculate Nash Equilibrium of route R	
p_i	average ticket price of A_i for route R
c_i^{psg}	unit passenger related cost of A_i in route R
$N(A_i)$	utility (profit) of A_i in route R
Symbols related to Profit Maximization of airline A	
C_i	per flight cost of route R_i for a given airline A
f_i	flight frequency of route R_i for a given airline A
$r_i(f_i)$	revenue when A operates f_i flights on route R_i
$N_i(f_i)$	profit (revenue minus total cost) when A operates f_i flights on route R_i
$\langle f_i \rangle$	frequency strategy for all routes in $\mathcal{R}$ of airline A
$\langle f_i^* \rangle$	optimal frequency strategy of airline A
$N(\langle f_i \rangle)$	profit of frequency strategy $\langle f_i \rangle$
b	budget limit of the Profit Maximization problem
K	the number of routes to optimize profit of airline
Symbols related to MAP-BBB in Algorithm 2	
n_p	parent node
c	child node
c_l, c_r	left and right children of a parent node
N^*	optimal profit sought by the algorithm
$MaxLB$	bounding threshold
UB_c, LB_c	lower bound and upper bound profit of node c
C_c	lower bound cost of node c
Symbols related to MAP-G in Algorithm 3	
g	a group of route
q	the number of routes in a group g
$\mathcal{G}$	a set of route groups, $\mathcal{R}$ is divided into $ \mathcal{G} $ groups
δ	budget allocation unit
$\langle f_g^* \rangle$	optimal frequency strategy of group g

TABLE 3.1: Summary of Notations

¹Like all past models, our analysis of airline profits is only based on publicly available data, so the true number might be smaller. Nevertheless, even a 10% improvement in the real-world would be substantial.

The rest of this chapter is organized as follows. Section 3.1 introduces existing work on airline market share and total demand prediction. Section 3.2 discusses the data set used in our study. Section 3.3 shows our proposed prediction framework based on our ensemble method. Section 3.4 presents our proposed algorithms for the profit maximization problem. Section 3.5 conducts extensive experimental evaluations. *For notational convenience, Table 3.1 summarizes all the notations used in this chapter.*

3.1 Existing Work on Airline Market Share and Demand Prediction

Existing airline market share and demand prediction models try to write down math formulas for market share/demand and then use regression to find values of the parameters in these formulas that minimize the sum of the squared errors between the predicted values and the true values. *We therefore call these **math models** to indicate that the structural form of these models is written down a priori without reference to any data and that the parameter values that minimize error are then computed using available data.*

Airline market share prediction. Most works on airline market share prediction use a multinomial logit (MNL) regression model [109–111]. For a set $\mathcal{A}$ of airlines in a given route, an airline A_i 's market share m_i is modeled as:

$$m_i = \frac{e^{V_i}}{\sum_{A_j \in \mathcal{A}} e^{V_j}}. \quad (3.1)$$

Here $V_i = \sum_k \eta_k X_{ik}$ is the customer's utility for choosing airline A_i 's service, X_{ik} is the value of the k^{th} variable for airline A_i and η_k is its corresponding weight to be learnt from data. Various variables have been studied. In Market1 [109], the authors consider a route's price, frequency, and the number of stops. Based on the prediction, they analyze competition among airlines. Market2 [110] considers an airline's frequency, delay, and safety. Market3 [111] studies the effect of aircraft size and seat availability on market share and considers other variables such as price and frequency.

Total demand prediction. For total demand of a route, a multiplicative [112–114] regression model is the most frequently adopted. Formally, a route’s total demand is represented as:

$$d = \prod_k Y_k^{\lambda_k}, \quad (3.2)$$

where Y_k is a value of the k^{th} variable and λ_k is a parameter to be learnt from data which measures the importance of Y_k . Demand1 [112] considers wealth related variables such as price, income and CPI (consumer price index). Demand2 [114] studies the influence of demographic variables such as price, income and population, as well as hub status (i.e., large, medium, small or non-hub) on passenger demand. Demand3 [113] focuses on variables such as price, income, population and origin-destination distance.

In this chapter, we show that our prediction models build on all of these methods, but use an ensemble approach to achieve not only much higher predictive accuracy, but also much greater robustness.

3.2 The MAP Dataset

We have created an integrated “MAP” data set by aggregating information from 4 publicly available data sources, viz. the Bureau of Transportation Statistics (BTS) [116], the Bureau of Economic Analysis (BEA) [117], the National Transportation Safety Board (NTSB) [118], and the U.S. Census Bureau (Census) [119]. BTS provides almost all aspects of airline market information such as ticket price¹, frequency, number of stops, delay records, aircraft size and available seats. The BEA and the Census release regional income, economic state, and population information while the NTSB provides all safety and accident related information. The BTS provides a quarterly dataset. A detailed description of the variables we consider is presented in Table 3.2. Our training data includes 10 years (40 quarters) of data on 13 airlines and 700 routes. Our validation data includes one quarter (all 700 routes and 13 airlines’ market share and profits

¹BTS releases a 10% sample of route and airline specific ticket sales logs, from which average ticket price is obtained.

are predicted, leading to a total 9100 predictions in total). This is consistent with most corporate forecasts where forecasts are made for the next quarter¹.

Market Share	
ticket price	average price of tickets sold by airline
frequency	number of flights operated by airline
delay time	average delay time of all flights operated by airline
delay ratio	ratio of delayed flights to total number of flights by airline
cancel ratio	ratio of canceled flights to total number of flights by airline
stop	average number of stops of all flights by airline
safety	number of casualties caused by past accidents of airline
craftsize	average number of seats for a flight of airline
total seat	total number of seats provided by airline during a quarter
Total Demand	
avg. price	average of all airlines' ticket prices
population	geometric mean of origin and destinations' population
income	geometric mean of origin and destinations' income
CPI	geometric mean of origin and destinations' CPI
distance	distance between origin and destination

TABLE 3.2: Variables used to make predictions by related works

3.3 Ensemble Prediction Model

In this section, we design an ensemble-based predictor MAP-EF that produces highly accurate route-specific predictions.

Overall architecture. Figure 3.1 shows the architecture of our ensemble model which runs through 4 major steps.

Step 1 - Past Models with Regularization. In the first step, we use the MAP-dataset, in conjunction with existing market share models (Market1 [109], Market2 [110], Market3 [111]) and demand models (Demand1 [112], Demand2 [114], Demand3 [113]) in order to generate the predictions made by models developed previously in the literature. These predictions are used as additional features. In addition, we augment past models

¹These predictions exceed that of related work, e.g., [111], which predicted for 10 airlines with a 10 year history for 200 routes.

Figure 3.1: Structure of the MAP-EF prediction model. Rectangles mean data and ellipses algorithms. Each algorithm is marked with a processing sequence number. Solid lines mean inputs and dashed lines outputs.

with regularization [120].¹ Because this step is straightforwardly built on top of past works, we do not go into it in further detail.

Step 2 - Nash Equilibrium. Ticket pricing clearly influences both market share and demand. At the same time, the competitive pricing behavior among airlines would have further influence. Thus, we include equilibrium pricing calculated based on past models as an additional feature in MAP-EF. At the end of this step, each route has one feature table for market share prediction and another for demand prediction. Each table² includes a set of features collected from the MAP-dataset and from Steps 1 and 2.

Step 3 - Clustering of Similar Routes. We cluster all routes into groups of similar routes with the features of the quarter we predict. For each group of routes, we concatenate all

¹In past works for market share prediction, the authors minimize sum-squared-error (SSE), i.e. they try to find parameter values for the math models that minimize $\sum_{A_i} (m_i - \hat{m}_i)^2$ where m_i (resp. $\hat{m}_i$) is the true (resp. predicted) market share. In addition to this, regularization minimizes $\sum_{A_i} (m_i - \hat{m}_i)^2 + ||W||$, where W is a vector that consists of parameters η_k (or λ_k for demand prediction) and $||W||$ is the norm of W . $||W||$ can be defined using either the L^1 or L^2 norm. Thus, we have three options: SSE, SSE+ L^1 regularization, and SSE+ L^2 regularization. In order to decide the best model among these, we use grid search and cross-validation. For cross-validation, we predict a quarter of the training data after training with the remaining quarters regardless of temporal sequence, which creates n -fold cross validation, where n is the number of quarters in the training data.

²Please refer to the Appendix in Section 6.5 for detailed examples of the merged table.

the individual route tables constructed in Step 2 for all routes in that group. The merged tables are then used for training within MAP-EF. We describe this step in further detail.

Step 4 - Iterative Gaussian Process Regression (GPR) and Random Sample Consensus (RANSAC). At the end of Step 3, we have a comprehensive *MAP Feature Table*. We use a combination of the well-known methods of GPR [121] and RANSAC [122]. To make market share/demand predictions for a route R , this step trains only on data about the group that contains route R , removes outliers using RANSAC, and then uses GPR to make the final prediction.

Nash equilibrium price feature. In a competitive airline market, the pricing behaviors of airlines are strategic. To capture the competitive nature of the market, we include equilibrium price as an additional feature¹. The rationale behind this is that when an airline's quoted price is higher/lower than its equilibrium price, the customer might lose/gain utility, which has influence on the customer's choice and thus further affects an airline's market share. Therefore, we use the gap between equilibrium price and real price as an additional feature. The utility (net income) $N(A_i)$ of an airline A_i w.r.t. a specific route is $N(A_i) = (p_i - c_i^{psg}) \cdot d \cdot m_i$, where p_i is the ticket price and c_i^{psg} is the passenger-related per ticket cost obtained from the BTS dataset. d and m_i are a route's total demand and airline A_i 's market share as described in Section 3.1. Recall that Nash equilibrium is a state in a multi-player non-cooperative game, where each player is assumed to know the equilibrium strategies of the other players, and no player has anything to gain by changing only its own strategy. Following this definition, we formulate the computation of Nash equilibrium strategy of each airline $A_i \in \mathcal{A}$ w.r.t. a given route:

$$\max_{p_i} \quad N(A_i), \quad (3.3)$$

$$\text{subject to} \quad p_i^{\min} \leq p_i \leq p_i^{\max}, \quad (3.4)$$

$$d \cdot m_i \leq Q_i. \quad (3.5)$$

¹Note that when computing the equilibrium price, the market share and total demand is the math formula obtained in Step 1, and the competition is only within a route.

Algorithm 1: ITER

```

1 Randomly assign values to  $\mathbf{p}_{t-1}$  within  $[p_i^{min}, p_i^{max}]$ 
2 repeat
3   | algorithm
4 until  $counter = n$ 
5 for  $A_i \in \mathcal{A}$  do
6   |  $counter \leftarrow 0$ 
7   |  $p_{i,t} \leftarrow$  Solution of Eqs.(3.3)-(3.5)
8   | if  $|p_{i,t} - p_{i,t-1}| \leq \varepsilon$  then  $counter++$ 
9  $\mathbf{p}_{t-1} \leftarrow \mathbf{p}_t$  return  $\mathbf{p}_t$ 

```

Eq.(3.4) specifies the feasible section of price, and Eq.(3.5) indicates that the total number of passengers cannot exceed its capacity Q_i . To compute equilibrium price on a route, we propose ITER (Iterative meThod for nash EquilibRium computation) in Algorithm 1, where $\mathbf{p}_{t-1}$ and $\mathbf{p}_t$ denote the optimal price vectors of two subsequent “virtual” periods. The algorithm starts with an initialization of $\mathbf{p}_{t-1}$ within $[p_i^{min}, p_i^{max}]$, where p_i^{min} and p_i^{max} are the minimum and maximum possible prices quoted by airline A_i . After this, iteration proceeds. In each repeat loop (Lines 2-9), the algorithm updates optimal prices of all the airlines. Within each repeat loop, there is a for loop (Lines 3-7). In the for loop, each airline alternately updates its optimal price $p_{i,t}$ based on the previous price vector $\mathbf{p}_{-i,t-1}$ of the other airlines in the last virtual period. It iterates until for each airline $A_i \in \mathcal{A}$, the price is not updating in two subsequent virtual periods (i.e., the difference between the two prices is no larger than a small threshold $|p_{i,t} - p_{i,t-1}| \leq \varepsilon$), which means a Nash equilibrium is obtained.

Theorem 3.1. *If ITER returns a solution, then that solution is the Nash equilibrium price.*

Proof sketch. In each repeat loop, ITER finds the best response of all players (airlines) given the strategies of the last virtual period. When the best response strategies of all players in the current virtual period are equal to the best response strategies of the last virtual period, a Nash equilibrium is found according to the definition of Nash equilibrium. $\square$

Clustering similar routes. Some routes are more similar than others, e.g., LAX-JFK may show more similar market attributes to SFO-JFK than LAX-SFO because the first

two are long-haul routes from west to east and the last one is a short-haul local route, and estimating market share/demand would probably benefit from examining all of similar routes. In order to cluster routes, we use all features gathered in the MAP-dataset, Steps 1 and 2. As we do not know the “ground truth” definition of whether a cluster is correct or not, we evaluate cluster quality via *silhouette score*¹ [123]. The silhouette score tends to keep increasing (with some glitches) as the number of clusters increases. The elbow method [124] chooses the optimal number of clusters when silhouette scores stabilize. We tested several clustering methods including K-Means⁺⁺ [125] and connectivity-based and density-based clustering algorithms such as DBSCAN [126]. Of these, we achieved the best result with K-Means⁺⁺.

RANSAC/GPR regression. Training with similar routes does not always contribute positively to predictive accuracy. In several cases, it incurs additional error. We tested many linear regression algorithms such as Ridge, Lasso, and Lars [120]. None of them could produce stable performance across multiple routes. We then tested a specially devised structure that consists of RANSAC and GPR. RANSAC removes outliers from training data and GPR makes the final prediction only with inliers. GPR is known to be robust because it is much less parametric than the others. Intuitively, GPR says that if features of two airlines are similar, their market shares should also be similar. It assumes that priors follow a Gaussian distribution. After that, GPR also calculates posteriors using the famous bayes rule, $\mathbb{P}(\text{posterior}) \propto \mathbb{P}(\text{likelihood})\mathbb{P}(\text{prior})$ that can be analytically calculated using a covariance matrix fitted to the training data. To predict, GPR, using the learned covariance matrix, calculates similarities between a test case and all training cases, and makes a prediction. The entire process can be described by a very complicated linear algebra form. RANSAC Regression is known to be robust w.r.t. noise. In the first step, RANSAC performs regression with a small random subset of training data. In the second step, all training data that does not fit the model constructed during the first step is classified as outliers and others as inliers. These two processes are iterated over time to refine the set of inliers called the *consensus*

¹The silhouette score is to calculate the consistency of clusters. As more similar routes are contained by a cluster, its silhouette score is higher. If $\mathbf{G}$ is a set of clusters, dis is some distance (dissimilarity) measure, and e is some data item, the silhouette score $SS(e) = \frac{b(e) - a(e)}{\max\{a(e), b(e)\}}$ where $a(e)$ is the average distance between e and the data items of the cluster it belongs to, and $b(e)$ is the minimum average distance between e and the data items of clusters it does not belong to.

set. Thus, MAP-EF uses RANSAC with GPR as a base estimator. We show prediction results of other regression techniques in the Appendix in Section 6.5.

Time Overheads. In comparison with past models, MAP-EF should spend additional time for clustering and more advanced regression methods, which takes cubic time of training data size.

3.4 Airline Profit Maximization via Frequency Allocation

In this section, we suggest how to maximize an airline's profit by optimally allocating quarterly frequencies to routes.

Definition 3.2 (Frequency Strategy). A frequency strategy of an airline A for a given set $\mathcal{R} = \{R_1, \dots, R_n\}$ of routes is a vector $\langle f_i \rangle$ of non-negative integers.

Intuitively, the i 'th element f_i of the frequency strategy denotes the number of times the airline should fly route R_i in a quarter. For a given airline A , a set $\mathcal{R} = \{R_1, \dots, R_n\}$, per flight cost C_i for each route R_i , and total operation budget b , the profit maximization problem finds the frequency allocation $\langle f_i \rangle$ of A assuming its competitors' behaviors are fixed. We also conduct experiments to test the robustness of this approach when competitors change their strategies simultaneously.

Definition 3.3. The **Frequency-based Profit Maximization Problem** can be encoded as the optimization problem:

$$\max_{\langle f_i \rangle} \sum_{R_i \in \mathcal{R}} N_i(f_i) \quad (3.6)$$

$$\text{subject to } 0 \leq f_i \leq f_i^{\max}, f_i \in \mathbb{N}, \forall R_i \in \mathcal{R} \quad (3.7)$$

$$\sum_{R_i \in \mathcal{R}} C_i f_i \leq b. \quad (3.8)$$

Figure 3.2: Quarterly revenue (dashed lines) and profit (solid lines) in 10 million dollars generated by MAP-EF. Revenue is defined as average ticket price multiplied by the number of passengers. Cost is per flight cost multiplied by frequency. Note that the profit-frequency curve (revenue minus cost) is very irregular even though the revenue-frequency curve is monotonically increasing. The shapes of the curves are quite different.

$N_i(f_i)$ is the profit of route R_i given frequency f_i , which is ticket revenue $r_i(f_i)$ minus total operation cost $C_i f_i$ and can be calculated by the market share and total demand predicted by MAP-EF. The ticket revenue is defined as price per ticket multiplied by airline A 's total passenger demand on route R_i . $\sum_{R_i \in \mathcal{R}} N_i(f_i)$ is the sum of profits for all routes. Because routes (such as the SFO-LAX and JFK-LAX routes shown in Figure 3.2) can differ dramatically in profitability, and as these numbers are basically estimated by our market share and demand prediction algorithms, it follows that the individual terms $N_i(f_i)$ in our objective function can vary dramatically in form from one route to another. This is what makes using a single form of objective function difficult. Constraint (3.7) specifies the bound of frequency for each route as f_i^{max} and that the frequencies are non-negative integers. The frequency bound $f_i^{max} = \min(\lfloor \frac{b}{C_i} \rfloor, f^{BTS})$, where f^{BTS} is the maximum frequency discovered in the BTS dataset, so that we prevent f_i from exceeding the feasible frequency bound.¹ Constraint (3.8) indicates the budget constraint. As discussed in Section 3.1, this optimization problem belongs to the hardest class of KPs and RAPs, mainly due to two reasons: i) the solution space is exponentially large as $\prod_i f_i^{max}$, and ii) the objective function is neither convex nor concave (thus not linear).

¹We can use larger upper bound values for f_i with a trivial modification to our algorithm.

Before presenting the complexity analysis and the algorithms to solve MAP, we introduce two important assumptions. First, we assume that varying the frequency of a route does not affect other routes' total demand and market share — *Independence of Routes*. This assumption is valid for most cases where the passenger's traveling pattern on one route is independent of the other routes. Second, we assume revenue (resp. cost) is a monotonically increasing non-linear (resp. linear) function of frequency — *Monotonicity of Revenue and Cost*. As shown in Figure 3.2, this assumption is intuitive. Increasing frequency leads to larger market share (thus higher revenue) and higher operation cost. Therefore, the monotonicity of revenue and cost does not imply the monotonicity of profit. This prevents us from applying many well-developed algorithms for KPs and RAPs. We show the computational hardness of the MAP problem with the following theorem.

Theorem 3.4. *The decision version of MAP (D-MAP) which answers the question “Is there a frequency strategy whose profit exceeds a threshold P without violating budget b ?” is NP-complete.*

Proof sketch. We first show that there is a polynomial reduction from the Knapsack problem to D-MAP. In the Knapsack problem, there are n items (each has p_i profit and w_i cost), budget limit W , a threshold V , and the objective is to find a set of items whose profit is no smaller than V without violating budget W . This is a special instance of D-MAP where $b = W$, $P = V$, $C_i = w_i$ and the profit-frequency function of route R_i is linear with the slope of p_i ; and the reduction from the Knapsack problem to profit maximization can be done in polynomial time.

Next, we show its correctness, i.e., Y is a *yes* instance of the Knapsack problem $\iff X$ reduced from Y is a *yes* instance for D-MAP.

“if” direction. Assume that there exists a solution s for a Knapsack problem instance. We can create a frequency strategy $\langle f_i \rangle$, where f_i is the number of i^{th} item in the solution s . s can achieve a profit of at least V without violating budget W , and so is $\langle f_i \rangle$ because $b = W$, $P = V$, $C_i = w_i$ and the same linear profit function.

“only if” direction. Assume that a frequency strategy $\langle f_i \rangle$ can achieve a profit of at least P without violating budget b . We can create a Knapsack solution s to select f_i

items for i^{th} item. s is a valid solution of the Knapsack problem because $b = W$, $P = V$, $C_i = w_i$ and the same linear profit function.

Finally, D-MAP is in $\mathcal{NP}$ because the verification process to check whether a frequency strategy's profit exceeds a threshold requires $|\mathcal{R}|$ oracle queries. Note that each oracle query takes a constant time for a given route and airline. Thus, the verification process takes polynomial time. $\square$

Due to the computational complexity, it is intractable to find an optimal solution of the MAP problem by performing brute-force search. To overcome these issues, we propose two new approaches: i) MAP-BBB is a bi-level branch and bound method, and ii) MAP-G is a greedy algorithm operating over groups of routes. Section 3.5 shows that these two algorithms have complementary performance in terms of runtime (scalability) and optimality.

3.4.1 Bilevel Branch and Bound (MAP-BBB)

Branch and bound (BB) is a popular search algorithm for combinatorial optimization to efficiently enumerate the entire solution space and find the optimal solution. The solution space of BB can be represented by a rooted tree and each child of a parent node considers a subset of solution space of its parent node. BB consists of a series of repeated branching and bounding processes. In the branching process, the feasible solution space is divided into a number of smaller subsets, while in the bounding process, an upper and a lower bound of the objective (airline profit in this problem) is derived for each subset of solution space. The largest lower bound is updated and, when the upper bound of a certain subset is smaller than the current largest lower bound, that subset is pruned since it does not contain the optimal solution. As we can see, the key of BB lies in finding a tight upper and lower bound for the bounding process, especially for upper bound. While most of the BB algorithms for Knapsack problems find upper bounds through linear relaxation [74], the complicated form of our prediction model prevents us from implementing such upper bounds. As a result, we propose a novel

Bilevel branch and bound (MAP-BBB) algorithm, which uses another sub-BB search to find an upper bound for the main BB process.

Algorithm 2 shows the MAP-BBB algorithm that builds on classical branch and bound methods. The search for an optimal solution explores a tree whose nodes are labeled with n intervals, one for each of the routes $R_1, \dots, R_n$ under consideration. Intuitively, if a node is labeled with the interval $[\ell, u]$ for a particular route R_i , it means that we are currently restricting the frequency in the interval $[\ell, u]$ for that route. MAP-BBB also maintains a priority queue Q of all the nodes where the head has the largest *upper bound* profit. The root node has the full range of $[0, f_i^{max}]$ for each route $R_i \in \mathcal{R}$. As we descend the tree, these intervals get smaller. In Figure 3.3, for example, the parent node n_p is divided into two children nodes c_l and c_r , where the frequency range of the last route of n_p is divided into two halves.

For the bounding process of child node $c \in \{c_l, c_r\}$, upper and lower bound profits of the optimal strategy that can be defined with the frequency range of c are calculated (to be further described below). If its lower bound LB_c is larger than the temporary maximum lower bound, denoted as $MaxLB$ in Algorithm 2, and its lower bound cost C_c of the optimal strategy does not violate the budget b , then we update $MaxLB$ (line 11). If its upper bound UB_c is no larger than $MaxLB$ or its lower bound cost C_c violates budget b , we prune this node (line 12). Once we find a node that consists of only non-divisible ranges, i.e., min and max of ranges are all the same, and its cost is within budget b , it is a candidate of the optimal frequency strategy. The optimal strategy is selected among them (line 13). N^* and $\langle f_i^* \rangle$ stand for the obtained optimal profit and optimal frequency strategy, respectively.

Lower bound profit and lower bound cost. Calculating a lower bound of profit LB_c and cost C_c of node c is a key to the MAP-BBB algorithm. We choose the min frequency value for each route's range and calculate its lower bound profit and cost with MAP-EF. This is valid because the optimal profit and the associated cost of node c are always no smaller than that.

Figure 3.3: An example branch and bound process for MAP-BBB with three routes. Each blue bar of node represents a frequency range. Lower BB is used to calculate an upper bound profit of a node. c_l and c_r are two children of the parent node n_p .

Upper bound. As we cannot employ linear relaxation, upper bound calculation is more challenging. For this, we use another branch and bound process, which is why the proposed algorithm is called a *bilevel branch and bound*. For each range of R_i in node c , we perform the following sub branch and bound process to find its optimal frequency within the range **without a budget constraint**. This is good because the optimal profit without budget limit is definitely no smaller than the optimal profit with budget constraint and it is much faster than the upper bound with budget check across multiple routes. We omit its pseudo code because it is more or less the same as Algorithm 2 except that the sub branch and bound does not check budget violation. In order to distinguish two branch and bounds, we use different terms *sub* and *main*. The sub root node is initialized with the specified range of R_i . During the sub branching process, we equally divide a sub node interval into two halves and create two sub children nodes. In the sub bounding process of a sub node, the heuristic for computing a lower bound of the sub branch and bound is defined as $r_i(f_{random}) - C_i f_{random}$, where f_{random} is a random frequency in the frequency range. This holds since any feasible frequency

Algorithm 2: Main branch and bound for MAP-BBB

```

1  $N^* \leftarrow 0, \langle f_i^* \rangle \leftarrow \mathbf{0}$ 
2 Create  $root$ 
3  $MaxLB \leftarrow LB_{root}$ 
4 PriorityQueue  $Q \leftarrow \emptyset$ 
5  $Q.enqueue(root)$ 
6 while  $|Q| > 0$  do
7    $n_p \leftarrow Q.poll()$ 
8   if  $n_p$  is divisible then
9     Create  $c_l, c_r$ 
10    for  $c \in \{c_l, c_r\}$  do
11      if  $LB_c > MaxLB$  and  $C_c \leq b$  then  $MaxLB \leftarrow LB_c$ 
12      if  $UB_c \geq MaxLB$  and  $C_c \leq b$  then  $Q.enqueue(c)$ 
13    else if  $N^* < N(n_p)$  and  $C_{n_p} \leq b$  then
14       $N^* \leftarrow N(n_p)$ 
15       $\langle f_i^* \rangle \leftarrow$  frequency strategy of  $n_p$ 
16 return  $\langle f_i^* \rangle, N^*$ 

```

generates a lower bound of the optimal profit. The heuristic for computing an upper bound of the sub branch and bound is $r_i(f_{max}) - C_i f_{min}$, where f_{max} and f_{min} are the maximum and minimum frequency of the sub node, which utilizes the monotonicity of revenue and cost. The sum of maximum profits calculated for each route with the sub branch-and-bound without budget constraint is always no smaller than that with a budget, thus sub branch-and-bound provides an upper bound profit of the main node. With the lower and upper bounds derived above, MAP-BBB eliminates only non-optimal frequency strategies of routes, which proves correctness and optimality of MAP-BBB. We show the best-case complexity analysis of MAP-BBB:

Theorem 3.5. *The best-case runtime complexity of MAP-BBB is $O((\log(\max\{f_i^{max}\}))^2 \times n)$, where n is the number of routes.*

Proof sketch. The best-case complexity of the bi-level branch and bound algorithm happens when in each branch and bound process, one of the children nodes is pruned. For each route R_i , its maximum frequency is f_i^{max} and it takes $\log(f_i^{max})$ branches until the frequency in this route is nondivisible; for all routes, it takes $\max\{f_i^{max}\} \times n$ time. While for each branch it needs at least $O(\log(\max\{f_i^{max}\}))$ time to solve one sub branch and bound. Thus, the total complexity is $O((\log(\max\{f_i^{max}\}))^2 \times n)$. $\square$

We need to clarify that f_i^{max} is the maximum frequency of a route R_i , while $\max\{f_i^{max}\}$ is the maximum frequency of all routes. The worst-case complexity happens when there is no pruning in each branch and bound process. However, the experimental results in Section 3.5 show that this never happens and the average runtime of MAP-BBB is promising.

3.4.2 Greedy Algorithm (MAP-G)

As MAP is NP-hard, it is clear that the exact algorithm MAP-BBB may not be scalable. In this section, we propose a greedy algorithm to solve MAP more efficiently while sacrificing optimality. A naive greedy algorithm is to increase the frequency of the route with the largest marginal profit/cost ratio (PCR) which is known to work very well for conventional knapsack problems. However, unlike conventional knapsack problems where PCR is fixed (i.e., linear objective function), the PCR in our problem fluctuates as can be seen from the example in Figure 3.4. According to our experiments, this naive greedy approach on average achieves only 0.66 of the optimal profit.

To address the fluctuating PCRs, we suggest an improved “Group-Greedy” algorithm (MAP-G). The basic idea of MAP-G is that, instead of greedily raising the frequency of a single route with the largest PCR, MAP-G first divides routes into groups, and then compares the PCRs of different groups and raises frequencies of the group of routes with the largest PCR. As shown in Algorithm 3, it randomly divides n routes into $\lfloor \frac{n}{q} \rfloor$ groups (line 2), each consisting of q routes¹. Initially, each group is assigned small budget δ (line 3) and we independently solve MAP within each group, using exact algorithms such as MAP-BBB as a sub-routine (line 6). We merge all groups’ most recent frequency strategy to create a complete frequency strategy of all routes (line 8), and its profit is compared with the temporary maximum profit (line 10). The budget of the group that creates the largest marginal profit density is increased by δ (line 9). This process iterates until the sum of groups’ budgets equals to the total budget b . This method is robust w.r.t. the fluctuating PCR because we calculate the optimal strategy for each group in each iteration. On the other hand, calculating optimal frequency strategy

¹The actual number of routes in a group may slightly vary due to the floor operation, while the manner of grouping different routes is of little relevance to the algorithm

Algorithm 3: MAP-G

```

1  $N^* \leftarrow 0, \langle f_i^* \rangle \leftarrow \mathbf{0}$ 
2  $\mathcal{G} \leftarrow$  divide  $\mathcal{R}$  into  $\lfloor \frac{n}{k} \rfloor$  groups
3  $b_g \leftarrow \delta, \forall g \in \mathcal{G};$ 
4 while  $\sum_{g \in \mathcal{G}} b_g \leq b$  do
5   for  $g \in \mathcal{G}$  do
6      $\langle f_g^* \rangle \leftarrow$  solve MAP of  $g$  with budget  $b_g$ 
7      $g^* \leftarrow$  the best PCR group
8      $\langle f_i \rangle \leftarrow \text{merge}(\{\langle f_g^* \rangle | g \in \mathcal{G}\})$ 
9      $b_{g^*} \leftarrow b_{g^*} + \delta;$ 
10    if  $N^* < N(\langle f_i \rangle)$  then  $\langle f_i^* \rangle \leftarrow \langle f_i \rangle$ 
11 return  $\langle f_i^* \rangle, N^*$ 

```

Figure 3.4: Best Profit-Cost Ratio among all routes in each iteration of the naive greedy Knapsack algorithm.

for a group of routes is much faster than that of all routes because the solution space is much smaller. We set $\delta = b \times \frac{1}{h}$, where $h > 1$ is an integer. The quality of the solution increases as q increases — if $q = n$ and $h = 1$, it finds the optimal solution. The benefit of a small q is the short runtime, while sacrificing optimality. By adjusting q , MAP-G can trade-off between runtime and optimality.

3.5 Experimental Results

In this section, we first compare MAP-EF with several benchmark methods for both route demand and airline market share prediction. We then show the improvement of the optimal profit obtained based on MAP-EF prediction compared with that based on traditional methods. Last, we compare both scalability and optimality of our proposed algorithms with several benchmarks.

3.5.1 Market Share & Total Demand Prediction

3.5.1.1 Experimental Environment

For route-specific predictions, we selected the 700 largest routes in terms of the number of passengers. We predicted market share and total demand for the first quarter of 2015, the most recent period for which all data are available, after training with 10 years data preceding 2015. We tested 3 pre-existing prediction models for both market share and total demand as described in Section 3.1, and our proposed method MAP-EF.

For each route, we evaluated with the following three criteria: i) Correlation Coefficient (CC), ii) R^2 , and iii) Mean Absolute Error (MAE) divided by the maximum market share value of a route or divided by true total demand value. These two metrics are much stricter than pure MAE values. A good predictor will perform well under all three metrics.

3.5.1.2 Experiment Results

Market Share. Table 3.3 (resp. Table 3.4) summarizes mean values (resp. variances) of the three metrics for all routes (All) and for routes with 4 or more operating airlines ($|\mathcal{A}| \geq 4$) — as the number of airlines increases, predictions become more challenging. Among existing models, Market1 shows the best performance. Its CC of 0.82 is quite reasonable with low variance around 0.18, but its poor R^2 of -0.84 indicates some issues. Two metrics revealing contradictory results indicates that the rise/decline of the predicted values may have high error. Thus, Market1 can tell us which airline has larger

market share than another, but its absolute market share value is not reliable. Profit optimization based on the best competitor, Market1, does not make sense because even a small error in predicting market share may lead to a high loss in profit. As shown in Table 3.3, MAP-EF shows very stable performance in terms of both mean and variance for all metrics, especially when $|\mathcal{A}| \geq 4$.

	All			$ \mathcal{A} \geq 4$		
	CC($\uparrow$)	$R^2(\uparrow)$	MAE /Max($\downarrow$)	CC($\uparrow$)	$R^2(\uparrow)$	MAE /Max($\downarrow$)
Market1	0.82	-0.84	0.133	0.82	0.18	0.157
Market2	0.48	-3.51	0.188	0.62	-1.92	0.212
Market3	0.34	-6.96	0.27	0.44	-3.99	0.32
MAP-EF	0.96	0.88	0.052	0.95	0.89	0.046

TABLE 3.3: Mean values of CC, R^2 and MAE/Max for market share predictions. $\uparrow$ indicates that larger values are preferred and vice versa. The best method is highlighted in yellow.

	All			$ \mathcal{A} \geq 4$		
	CC($\downarrow$)	$R^2(\downarrow)$	MAE /Max($\downarrow$)	CC($\downarrow$)	$R^2(\downarrow)$	MAE /Max($\downarrow$)
Market1	0.18	132.2	0.06	0.09	4.29	0.08
Market2	0.54	109.7	0.08	0.31	26.77	0.09
Market3	0.55	964.9	0.10	0.35	38.66	0.10
MAP-EF	0.03	0.12	0.01	0.01	0.05	0.01

TABLE 3.4: Variance of CC, R^2 and MAE/Max of Market Share Predictions.

Total Demand. Whereas route-specific market share prediction is a list (one value for each airline), total demand prediction of a route is a scalar value. Thus, we create two lists, one of true demand and one of predicted demand, and perform CC and R^2 analysis. MAE divided by true demand can be measured for each route and its mean and variance are also summarized in Table 3.5. Again, MAP-EF shows better performance than all other methods.

All these results indicate that profit maximization should be based on the prediction results of MAP-EF, while “optimal” strategies obtained with other models are more likely to be far from real optimal strategies. Nevertheless, we note that MAP-EF builds upon other pre-existing prediction models’ findings and hence we build on top of these prior scientists’ work.

	CC($\uparrow$)	R^2 ($\uparrow$)	MAE/True (Mean, $\downarrow$)	MAE/True (Variance, $\downarrow$)
Demand1	0.77	0.31	0.29	0.06
Demand2	0.49	-1.4	0.57	0.1
Demand3	0.56	-0.8	0.49	0.08
MAP-EF	0.98	0.96	0.07	0.004

TABLE 3.5: CC, R^2 and MAE divided by true demand of Total Demand Predictions.
Note that the variance of CC and R^2 cannot be defined because it predicts a scalar value for each route.

3.5.2 Profit Maximization

3.5.2.1 Experimental Environment

We performed experiments on a cluster of 64 machines running Linux with 2.4GHz Xeon CPU and 24GB RAMs. We selected six major airlines and calculated the optimal frequency allocation strategy for their top K biggest markets (a market means an origin-destination pair, i.e., a route). We retrieved real budget information of routes from the BTS dataset such that budget for a set of routes is the sum of their real budgets. For the scalability test, we chose the number of markets $K = \{2, 3, 4, 5, 6, 7, 8, 9, 10, 20, 30\}$. We compare the proposed MAP-BBB and MAP-G with two benchmark algorithms, namely N-Greedy (naive greedy algorithm) and DP (dynamic programming). DP is a well-known approach for KPs and RAPS and we refer to Appendix in Section 6.5 for a detailed description of DP. MAP-BBB is the proposed bilevel branch-and-bound algorithm. We used MAP-BBB as subroutine for MAP-G to solve the profit maximization subproblem in each group.

3.5.2.2 Maximum Profit of Different Predictors

Ground truth is not available for the true maximal profit an airline could possibly make as their actual quarterly profit may not reflect the total profit that they could have made. In theory, MAP-BBB is guaranteed to produce the maximal profit *under the assumptions made in our model*. As we see below, MAP-BBB works when the number of markets is relatively small. In order to explore the profits obtained using predictions made by pre-existing market share predictors [109–111], we compare them with the

Figure 3.5: Profit maximization with different prediction methods (relative to MAP-EF) and 90% confidence interval is shown.

profits obtained using the MAP-EF predictor. We also obtain the optimal frequency allocation strategies using MAP-BBB for $K \leq 6$ based on different prediction methods, and then calculate the corresponding profit of these strategies using MAP-EF (for larger K , MAP-BBB, as an exact algorithm, takes an inordinate amount of time). As shown in Figure 3.5, the optimal profit obtained by all existing methods has a gap of around 30% compared with MAP-EF, which is a huge amount considering the scale of profit (typically on the order of millions of dollars per route per quarter).

3.5.2.3 Comparing Different Optimization Algorithms

In this subsection, we first compare several algorithms which are targeted at small K (number of markets) values, such as DP, MAP-BBB, and N-Greedy. Using these algorithms as subroutines, we then tested the performance of MAP-G with large K values. Average optimized profits of airlines and runtime are summarized in Figures 3.6 and 3.7. “Real” stands for the average profit of airlines calculated with real frequency strategies in the BTS dataset. Interestingly, “Real” shows worse results than even the simple N-Greedy. We see that past market share models can only capture about 75% of the profit generated by MAP-EF. This is a substantial loss in absolute dollars.

Profit and Runtime Analysis for $K \leq 6$. As we can see from Figure 3.6 (a), since both DP and MAP-BBB are exact algorithms, their maximum profit values are the same and

Figure 3.6: (a) Optimized profits and (b) Runtime for $K \leq 6$.

are the largest. However, DP's runtime, shown in Figure 3.6 (b), grows exponentially and we could not finish it on time for $K \geq 4$. N-Greedy cannot achieve a reasonable profit. When $K = 2$, it has around 70% of the optimal profit calculated with exact algorithms; as K increases, however, it decreases to only 60%. Before $K = 5$, MAP-BBB can find the optimal profit in a few seconds while DP takes around 10,000 seconds in average. This is due to its pseudo polynomial runtime, which behaves more like exponential runtime than polynomial time. This indicates that MAP-BBB scales much better than other baselines and is more suitable as the subsolver of MAP-G.

Profit and Runtime Analysis for $K \geq 5$. Table 3.6 shows the optimality of MAP-G when $K = 4, 5$ or 6. The exact maximum profit when $K \geq 7$ cannot be computed as the exact algorithms either run out of memory or take inordinately long. But in cases when $K \leq 6$, MAP-G achieves 99% of the maximum profit suggesting that it does very well. As expected, MAP-G is robust w.r.t. the problem of fluctuating profit and independent processing of each route from which N-Greedy suffers. Figure 3.7 (a) shows that MAP-G is consistently better than N-Greedy and N-Greedy obtains only about 70% of MAP-G's profit, suggesting that MAP-G is far superior. We confirm that $q = 3$ (i.e. groups of similar routes are limited to have 3 routes in them) leads to an increase in profit than when $q = 2$ for MAP-G — in Figure 3.7 (a), MAP-G with $q = 3$

Figure 3.7: (a) Optimized profits and (b) Runtime for $K \geq 5$,

	$K = 4$	$K = 5$	$K = 6$
Profit	0.99	0.97	0.99

TABLE 3.6: Profit maximization of MAP-G(BBB) with $q = 3$ (relative to the optimal profit) when $K = 4, 5$ or 6 . K is the total number of routes, q is the number of routes allowed in a group.

is consistently better than $q = 2$ — which corresponds to our conjecture that optimality will be improved as the size of groups increases.

The profit increase from “Real” to the optimized one is quite impressive. Airlines can increase their profits by at least 55% by adopting MAP based on our model with available data. But please note that we assume the behaviors of other airlines are fixed, which is not the case in reality. In the following, we will test the robustness of MAP, i.e., when the frequency and pricing strategies of other airlines also vary (and not known in advance), how much profit can be achieved by the airline to be optimized.

Robustness Analysis. In the above experiments, we assume that the frequency allocation and pricing strategies of all other airlines are fixed and known in advance. We now test the robustness of our approach, i.e., we test the airline profits obtained by our approach when the frequency and pricing strategies of other airlines vary. To do this, we randomly varied price (resp. frequency) by a rate in $\{[-20\%, -10\%), [-10\%, 0),$

0, $[0, 10\%)$, $[10\%, 20\%]$. We first compute the optimal strategy by assuming that the other airlines' behaviors are fixed (because they are unknown). We then evaluate this obtained optimal strategy with the prediction model of the changed price and frequency strategies of other airlines. We computed the ratio of the sub-optimal profit (not knowing other airlines' price/frequency changes) to the optimal profit (knowing the changed price/frequency strategies of other airlines) — in order to calculate the exact optimal profit we tested K up to 6. For each frequency and price change rate combination, we generated 30 scenarios with all different random seeds and profits are averaged on them. As is shown in Figure 3.8, this profit ratio is always larger than 0.8, which indicates robustness of our MAP approach even with a 40% fluctuation of both price and frequency. Moreover, it is shown that for a fixed frequency change rate, the profit ratio is almost the same for all price change rates, which means that frequency change is more effective in airlines' competition. Last, the largest ratio happens in the case that competitors increase frequencies, which means MAP is more effective at competitive markets.

Figure 3.8: Profits by varying price and frequency of other competitors

3.6 Chapter Summary

In this chapter, we study the problem of large scale, data-driven decision making in a single agent environment, which is applied to a frequency-based airline profit maximization problem. We propose the MAP framework for optimal frequency allocation over multiple routes, with three key contributions. i) We design a novel ensemble predictor based on past regression-based work, clustering techniques and game theoretic

analysis, which have never been utilized for this purpose; ii) based on the prediction, we design several algorithms to solve the profit maximization problem; iii) we conduct extensive experiments. We show that the prediction performance of MAP-EF is much better than past methods. We also compare runtime and optimality of our proposed optimization algorithms with benchmarks. We also show that MAP increases profitability per route by at least 55%. Even if there are factors we did not consider on the basis of open source data and these numbers are reduced substantially if private airline data was to be used, the increased profitability would still be substantial.

3.7 APPENDIX

Feature Tables Generated after Step 2 of MAP-EF

Tables 3.7 and 3.8 separately present the example tables of variables for market share and total demand prediction, which are generated in Step 2 of MAP-EF. We take Table 3.7 as an example to illustrate. In this table, the first 4 columns (“Route, Quarter, Airline, Existing Variables”) are existing features which are also used by past works. Column 5 is the output in Step 1 of MAP-EF, including the parameter values η_k in the math model and the intermediate predicted market share values. Column 6 is the difference between Nash equilibrium price and the real price of an airline, which is obtained in Step 2. The highlighted last column (“True market share”) indicates the dependent variable which is to be predicted. Based on both the market share related features extracted in Table 3.7 and the demand related features in Table 3.8, we further employ clustering techniques in Step 3 to find similar routes.

Route	Quarter	Airline	Existing Variables	Outputs of Step 1	Equil. Price — Real Price	True Market Share
R_1	2010Q1	A_1	Price, Freq, Delay, ...	15% (Predicted value), η_k (Parameters)	-50	10%
R_1	2010Q1	A_2	Price, Freq, Delay, ...	85%, η_k	60	90%
R_2	2010Q1	A_1	Price, Freq, Delay, ...	11% (Predicted value), η_k (Parameters)	-10	5%
R_2	2010Q1	A_2	Price, Freq, Delay, ...	89%, η_k	50	95%
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

TABLE 3.7: Variables to predict airline market share of R_1 by MAP-EF. “True Market Share” (highlighted) is the dependent variable to be predicted.

Route	Quarter	Existing Variables	Outputs of Step 1	Equilibrium Price	True Total Demand
R_1	2010Q1	Avg Price, Population, Income, ...	20,000 (Predicted value), λ_k (Parameters)	\$340	22,000
R_1	2010Q2	Avg Price, Population, Income, ...	25,000, λ_k	\$350	24,000
R_2	2010Q1	Avg Price, Population, Income, ...	19,000 (Predicted value), λ_k (Parameters)	\$360	21,000
R_2	2010Q2	Avg Price, Population, Income, ...	18,000, λ_k	\$250	20,000
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

TABLE 3.8: Variables to predict total demand of R_1 by MAP-EF. “True Total Demand” (highlighted) is the dependent variable to be predicted.

Prediction Results of other Regression Techniques

Tables 3.9 and 3.10 show the market share prediction performances of all tested regression algorithms including MAP-EF(Ridge), MAP-EF(Lasso), MAP-EF(Lars), and MAP-EF(TheilSen). We refer to Section 3.5.1 for a description of the experiment settings and the three metrics (CC, R^2 , MAE/Max) for measuring prediction performance. As we can see in Table 3.9, our selected MAP-EF with RANSAC and GPR (herein we call it MAP-EF for short) shows the best performance in the mean values of all metrics. While the mean metrics values of some regression algorithms are quite close to MAP-EF(e.g., MAP-EF(Lars)), it is shown in Table 3.10 that their variances are much larger than MAP-EF, which means others regression algorithms are not as stable as MAP-EF. This is the main reason why we choose MAP-EF.

Table 3.11 shows the total demand prediction results for all the regression algorithms. As we can see, the mean values of both CC and R^2 of MAP-EF are larger than the other regression algorithms, while the mean and variance values of MAE/True are much smaller. This also indicates the superiority of MAP-EF while predicting total demand.

Dynamic Programming Approach

The following recurrence formula shows the suggested dynamic programming approach:

$$N^*(i, w) = \max_{k=0}^{f_i^{max}} \{N^*(i-1, w - k \cdot C_i) + N_i(k)\},$$

	All			$ \mathcal{A} \geq 4$		
	CC($\uparrow$)	$R^2(\uparrow)$	MAE /Max($\downarrow$)	CC($\uparrow$)	$R^2(\uparrow)$	MAE /Max($\downarrow$)
Market1	0.82	-0.84	0.133	0.82	0.18	0.157
Market2	0.48	-3.51	0.188	0.62	-1.92	0.212
Market3	0.34	-6.96	0.27	0.44	-3.99	0.32
MAP-EF(Ridge)	0.92	0.65	0.07	0.91	0.64	0.07
MAP-EF(Lasso)	0.93	0.71	0.06	0.93	0.69	0.06
MAP-EF(Lars)	0.94	0.71	0.06	0.91	0.70	0.06
MAP-EF(TheilSen)	0.82	0.13	0.07	0.81	0.70	0.07
MAP-EF	0.96	0.88	0.052	0.95	0.89	0.046

TABLE 3.9: Mean values of CC, R^2 and MAE/Max for market share predictions. $\uparrow$ indicates that larger values are preferred and vice versa. The best method is highlighted in yellow.

	All			$ \mathcal{A} \geq 4$		
	CC($\downarrow$)	$R^2(\downarrow)$	MAE /Max($\downarrow$)	CC($\downarrow$)	$R^2(\downarrow)$	MAE /Max($\downarrow$)
Market1	0.18	132.2	0.06	0.09	4.29	0.08
Market2	0.54	109.7	0.08	0.31	26.77	0.09
Market3	0.55	964.9	0.10	0.35	38.66	0.10
MAP-EF(Ridge)	0.06	2.66	0.02	0.05	3.07	0.03
MAP-EF(Lasso)	0.06	2.31	0.02	0.04	3.2	0.02
MAP-EF(Lars)	0.06	2.33	0.02	0.04	3.1	0.03
MAP-EF(TheilSen)	0.23	18.54	0.02	0.21	0.13	0.02
MAP-EF	0.03	0.12	0.01	0.01	0.05	0.01

TABLE 3.10: Variance of CC, R^2 and MAE/Max of Market Share Predictions.

	CC($\uparrow$)	$R^2(\uparrow)$	MAE/True (Mean, $\downarrow$)	MAE/True (Variance, $\downarrow$)
Demand1	0.77	0.31	0.29	0.06
Demand2	0.49	-1.4	0.57	0.1
Demand3	0.56	-0.8	0.49	0.08
MAP-EF(Ridge)	0.34	-4.68	0.22	0.07
MAP-EF(Lasso)	0.94	0.92	0.12	0.02
MAP-EF(Lars)	0.95	0.93	0.12	0.02
MAP-EF(TheilSen)	0.95	0.88	0.14	0.01
MAP-EF	0.98	0.96	0.07	0.004

TABLE 3.11: CC, R^2 and MAE divided by true demand of Total Demand Predictions. Note that the variance of CC and R^2 cannot be defined because it predicts a scalar value for each route.

where $N^*(i, w)$ is the best profit we can get when operating flights from R_1 to R_i and the budget is w . $N_i(k)$ is the profit of operating k flights on R_i . Hence, $N^*(n, b)$ will output the optimal strategy of the profit maximization.

Example 3.1. Assume there are three routes, R_1 , R_2 , and R_3 ; and $C_1 = C_2 = C_3 = 10$ and $b = 100$. The optimal profit is $N^*(3, 100) = \max\{N^*(2, 100), N^*(2, 90) + N(3, 1), N^*(2, 80) + N(3, 2), \dots\}$.

Scatter Plots

Figure 3.9 shows full details of market share predictions, especially when $|\mathcal{A}| \geq 4$ (the most challenging prediction case). Market share prediction of a route is evaluated with three metrics: CC, R^2 , and MAE/Max. For each pair of possible combination of these metrics, we show how well each prediction model predicts in the figure. As is shown, MAP-EF's dots are always collected at the "good" corner of the chart. To illustrate, we take Figure 3.9(a) as an example, which shows the prediction performance of metrics R^2 vs. CC. As we know, larger R^2 and CC values indicate better prediction. In this figure, MAP-EF's dots always stay at the top right corner of the chart, which means it always has large R^2 and CC values. For the dots of existing prediction methods, while the dots of some routes stay at the top right corner, the rest of them do not. Worse still, some dots are far away from the corner. As is pointed out, even a small prediction gap can lead to a large profit loss in reality. These results indicate that market share prediction should be performed by MAP-EF.

Figure 3.9: Scatter plot of market share predictions when $|\mathcal{A}| \geq 4$. Each dot corresponds to a prediction of route.

Chapter 4

Strategic Decision Making in Multiagent Systems

In Chapter 3, we have studied the problem of large scale decision making in a single agent environment, where we assume that the strategies (i.e., decisions) of the other agents are fixed and do not change. In this chapter, we extend this type of problems to scenarios where the utility of an agent does not only depend on the decision made by itself, but also on the decisions of other agents. First, we take into account the existence of competition among agents, which is also applied to the airline profit maximization problem. We then consider a more complicated scenario where both competition and cooperation exist amongst a group of agents. This scenario is motivated by a practical problem related to the resource sharing and workload factoring in the cloud computing domain.

4.1 Frequency Allocation with Multiple Strategic and Competing Airlines

Previously, we assume that only one airline is strategic in deciding its frequencies of a certain set of routes $\mathcal{R}$, which may not be true in practice. In this section, we investigate the case where multiple airlines are strategic and change their frequencies using our

proposed frequency-based profit maximization algorithm and optimally respond to the current frequency strategies of the other airlines. The selfish and strategic behaviors of the competing airlines result in a non-cooperative game among the airlines, for which Nash equilibrium is a natural solution concept. Recall the definition of Nash equilibrium in Chapter 3, in a Nash equilibrium, the frequency strategies of the each airline $A \in \mathcal{A}$ satisfy

$$\max_{\langle f_{i,A} \rangle} \sum_{R_i \in \mathcal{R}} N_{i,A}(f_{i,A}), \quad (4.1)$$

$$\text{subject to } f_{i,A}^0(1 - \alpha) \leq f_{i,A} \leq f_{i,A}^0(1 + \alpha), f_{i,A} \in \mathbb{N}, \forall R_i \in \mathcal{R} \quad (4.2)$$

$$\sum_{R_i \in \mathcal{R}} C_{i,A} f_i \leq b_A. \quad (4.3)$$

We add a subscript A in f_i, f_i^0, C_i and b to distinguish different airlines, and propose ITER-Freq (ITER w.r.t. Frequency strategies) to compute Nash equilibrium frequency strategies, which is shown in Algorithm 4. First, we initialize the frequency of each airline A for each route R_i by randomly assigning a value within a bounded frequency range $[f_{i,A}^0(1 - \alpha), f_{i,A}^0(1 + \alpha)]$. In the repeat loop (Lines 2-8), it updates the optimal frequency strategies of each airline with the for loop (Lines 3-7) and terminates until that, for each airline A and for each route R_i , the difference between the updated frequency value $f'_{i,A}$ and the original value $f_{i,A}$ is within a small bound ε . Note that after each iteration, the predicted profit function (i.e., MAP-EF) is updated with the new frequency values of the corresponding airlines. To do this, the features described in Chapter 3 are also updated.

Theorem 4.1. *If ITER-Freq returns a solution, then that solution is the Nash equilibrium frequency.*

Proof. In each repeat loop, ITER-Freq finds the best response of all airlines given their strategies of the last iteration. When the best response strategies of all airlines in the current iteration are equal to the best response strategies of the last iteration, a Nash equilibrium is reached according to its definition. $\square$

The existence and uniqueness of Nash equilibrium heavily depend on the shape (e.g., concave, convex) of the profit functions of the competing airlines. Unfortunately,

Algorithm 4: ITER-Freq

```

1 Randomly assign values to  $\langle f_{i,A} \rangle$  within  $[f_{i,A}^0(1 - \alpha), f_{i,A}^0(1 + \alpha)]$ 
2 repeat
3    $counter \leftarrow 0$ 
4   for  $A_i \in \mathcal{A}$  do
5      $f'_{i,A} \leftarrow$  Solution of Eqs.(4.1)-(4.3)
6     if  $|f_{i,A} - f'_{i,A}| \leq \varepsilon$  then  $counter++$ 
7      $\langle f_{i,A} \rangle \leftarrow \langle f'_{i,A} \rangle$ 
8 until  $counter = n$ 
9 return  $\langle f_{i,A} \rangle, \forall A \in \mathcal{A}$ 

```

the profit functions are represented with a very complicated linear algebra form, and thus it is infeasible to analytically study the existence and uniqueness of Nash equilibrium frequencies. At the same time, it is known that the runtime complexity of iteration-based computation of Nash equilibrium is unpredictable, while in practice, it normally takes a few number of iterations before convergence of equilibrium (if there is).

4.1.1 Evaluation

In this part, we evaluate the convergence behavior and profits obtained with Nash equilibrium frequency. To do it, from the 5 largest airlines (denoted as A_1, A_2, A_3, A_4, A_5) in the US, we select 3 out of them which we assume are strategic and use Nash equilibrium frequency strategies, and then evaluate when these 3 airlines are competing over different number of routes ($K = 1, 2, 3$). Since there are in total 10 combinations of airlines ($C_3^5 = 10$), for each number of routes, all the following results are averaged over 10 cases. Table 4.1 shows the percentage of convergence¹ to equilibrium and, if a case converges, the averaged number of iterations required to reach the equilibrium (not counting the cases where it does not converge). For percentage of convergence, the number in the denominator means the number of cases reported. Note that for $K = 3$, this number is not equal to 10 because for one case, not all of the three airlines are operating over the same three routes simultaneously. We note that some cases do not converge to a Nash equilibrium within 10 iterations. However, for those cases which do

¹Note that we regard a case as not able to reach Nash equilibrium when the ITER-Freq algorithm does not converge within 10 iterations.

converge, they converge very quickly – 1.8 to 2.4 iterations on average. Our interpretation for the cases which do not converge is that, for these routes, the profit function of certain airline(s) obtained with our MAP-EF method is not concave, and the frequency values are restricted to natural number. We prove in the following theorem (Theorem 4.2) that, when the frequency domain is relaxed to continuous and the profit function of each competing airline is concave, Nash equilibrium always exists.

Theorem 4.2. *When the frequency domain of airlines (described in Eqs.(4.2)-(4.3)) is relaxed from natural number (i.e., $f_{i,A} \in \mathbb{N}$) to positive real number (i.e., $f_{i,A} \in \mathbb{R}^+$), and the profit function $\sum_{R_i \in \mathcal{R}} N_{i,A}(f_{i,A})$ of each airline A is concave w.r.t. its frequency strategy $\mathbf{f}_A$, Nash equilibrium (w.r.t. frequency) always exists.*

Proof. Denote the domain described in Eqs.(4.2)-(4.3) (with linear relaxation) as D . Accordingly, D_A denotes the domain for each airline A and D_{-A} denotes the domain for all the other competing airlines $A \in \mathcal{A}$ except A . For each airline A as a player, given the frequency strategies $\mathbf{f}_{-A}$ of all the other competing airlines, it will select an optimal frequency strategy which maximizes its profit function, which is a mapping $\phi_A : D_A \rightarrow D_{-A}$ defined as $\phi_A(\mathbf{f}_{-A}) = \mathbf{f}_A$. Furthermore, according to Eqs.(4.2)-(4.3), both the domain and range of the mapping $\phi_A(\mathbf{f}_{-A})$ are compact (closed and bounded). Combining the compactness of the domain and range of the mapping with the concavity of airline A 's profit function $\sum_{R_i \in \mathcal{R}} N_{i,A}(f_{i,A})$, it is guaranteed that the mapping $\phi_A(\mathbf{f}_{-A})$ is non-empty (*existence*). The compactness of D_{-A} along with the linear relaxation also implies that the mapping $\phi_A(\mathbf{f}_{-A})$ is *upper hemicontinuous*.

The existence and upper hemicontinuity of the mapping $\phi_A(\mathbf{f}_{-A})$ are naturally inherited by the mapping $\phi : D \rightarrow D$ defined as $\phi(\mathbf{f}) = \times_{A \in \mathcal{A}} \phi_A(\mathbf{f}_{-A})$, where $\mathbf{f}$ is a matrix which denotes the frequency strategies of all the competing airlines. In addition, Eqs.(4.2)-(4.3) (with linear relaxation) imply that D is a *simplex* (and closed). Applying Kakutani's fixed point theorem [35], we conclude that there always exists a fixed point $\mathbf{f}$ in the domain D which maps to a same point $\mathbf{f}$ in the range D with the mapping $\phi(\mathbf{f})$. As a result, with the above stated linear relaxation and concavity assumption, the Nash equilibrium (w.r.t. frequency) always exists. $\square$

It is worth mentioning that, in this problem, the concavity of airlines' profit functions is not a necessary condition (but only sufficient condition) of the existence of Nash equilibrium. Nonetheless, evidence (e.g., Figure 3.2 where profit airlines' functions of one route are displayed) shows that the profit functions of airlines are sometimes non-concave, which we think is a major reason why some cases do not converge to equilibrium within 10 iterations. However, it is still possible that Nash equilibrium still exists when the profit functions of some competing airlines are not globally concave, but for example, are locally concave.

K	1	2	3
Percentage of convergence	5/10	4/10	5/9
Number of iterations	1.8	2	2.4

TABLE 4.1: The convergence behavior of Nash equilibrium frequency.

airline \ K	1	2	3
A_1	[1.27,1.30]	[1.37,1.44]	[1.36,1.43]
A_2	[2.18,2.73]	[2.81,2.94]	[2.30,2.27]
A_3	[1.51,1.56]	[1.58,1.75]	[1.46,1.69]
A_4	[N/A]	[N/A]	[1.34,1.27]
A_5	[1.33,1.67]	[1.14,1.30]	[2.51,2.90]

TABLE 4.2: The profits obtained by airlines under equilibrium.

To evaluate the profits obtained by airlines with equilibrium strategy, we compare the profits of the airlines for three situations: 1) airlines use current frequency strategy; 2) only one airline is strategic; and 3) multiple airlines are strategic. As shown in Table 4.2, we show the profits obtained by the 5 airlines over different number of routes. For each cell in the table, the left number in the bracket denotes the ratio of profits of the second and first situations, while the right number denotes the ratio of profits for the third and first situations. Note that N/A means not applicable. This is because for $K = 1$ and $K = 2$, all the cases where airline A_4 is involved do not converge to equilibrium and thus we do not record the profits of these cases. All the profit ratios are averaged among the cases where equilibrium frequencies converge. For example, for airline A_1 and $K = 1$, if in 4 out of 6 ($= C_2^4$) cases the equilibrium frequencies converge, both the left and right profit ratio numbers in the for A_1 and $K = 1$ are averaged over 4 cases. As we can see, first, for all number of routes and all airlines, the profits obtained in both the

second (only one airline is strategic) and third (when multiple airlines are strategic) situations are larger than that obtained in the first situation (airlines use current frequency strategy). Second, it shows that, somewhat counter-intuitively, with a few exceptions (highlighted with yellow), the profits obtained when multiple airlines are strategic are mostly even larger than those obtained when only one airline is strategic. Note that this result is applicable under the assumption that each airline is only optimizing one-shot profit. When each airline is only concerned about short term profit, instead of “competing” with each other by increasing frequencies, airlines tend to “concede” and decrease their frequencies for routes that are non-profitable. This result, is a vast improvement that provides an insight to the researchers for a more precise profit prediction model which takes into account not only profits in the current period, but also profits in the future.

4.2 Workload Factoring and Resource Sharing via Joint Vertical and Horizontal Cloud Federation Networks

We have studied the case of decision making in a multiagent system where there is competition among a group of agents. In this section, we consider a more complex scenario where there are both competition and cooperation among the group of agents. In Chapter 1, we have introduced a typical motivation domain in cloud computing, where secondary cloud providers could benefit from forming a joint vertical and horizontal cloud federations. In this section, we present our proposed VHCF framework and solution algorithms.

4.2.1 Cloud Network System

In this subsection, we first introduce the secondary clouds and horizontal federation. We then describe the primary clouds and vertical federation. Last, we define the coalition values of the horizontal federations.

4.2.1.1 Secondary Cloud and Horizontal Federation

We consider a set of secondary clouds denoted as $\mathcal{J} = \{1, 2, \dots, j, \dots, |\mathcal{J}|\}$. Each secondary cloud j has a pool of computing resources. Each secondary cloud j is characterized by three parameters: the number of available resources (capacity) c_j^s , the number of requests r_j^s from its end users and the per unit utility p_j^s it gets for processing a unit user request.¹ We denote a horizontal federation, or *coalition* of a subset of secondary clouds as $C \subseteq \mathcal{J}$. Once a coalition C is formed, the resources and requests of its members are aggregated: $c(C) = \sum_{j \in C} c_j^s$ and $r(C) = \sum_{j \in C} r_j^s$. A *partition* of secondary clouds, or a *coalition structure* $\pi = \{C_1, \dots, C_k, \dots, C_m\}$, consists of different coalitions. We denote Π as the set of all coalition structures. Following [45] and [43], we restrict that a partition π is exhaustive and disjoint: $\cup_{k=1}^m C_k = \mathcal{J}$, and $C_k \cap C_{k'} = \emptyset$ (for any $C_k, C_{k'} \in \pi$ such that $C_k \neq C_{k'}$).

4.2.1.2 Primary Cloud and Vertical Federation

Instead of processing all the requests with the federated resource pool, a horizontal federation may also outsource requests to primary clouds via *vertical federation*. We denote the set of primary clouds as $\mathcal{I} = \{1, 2, \dots, i, \dots, |\mathcal{I}|\}$. Similarly, each primary cloud i is characterized by its capacity c_i^p , number of requests r_i^p and unit price p_i^p . Denote the number of requests from coalition C_k to primary cloud i as z_{ki} , the total number of requests received by primary cloud i is $r_i^p = \sum_{C_k \in \pi} z_{ki}$. The *request strategy* of coalition C_k is denoted as a vector $\mathbf{z}_k = \langle z_{k1}, \dots, z_{ki}, \dots, z_{k|\mathcal{I}|} \rangle^T$,² and the *request strategy profile* associated with a coalition structure $\pi = \{C_1, C_2, \dots, C_m\}$ is a matrix $\mathbf{z} = \langle \mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_m \rangle$.

¹We assume r_j^s is predictable and refer to [127] as an example of demand prediction. For ease of analysis, we consider only one type of computing resource, as is the case in [29]. Throughout the chapter, the superscript p means “primary”, while s means “secondary”.

²Note that this formulation is a generalization of “vendor lock-in”, which means that a secondary cloud can outsource requests to only one primary cloud. Nonetheless, our formulation can be reduced to the vendor lock-in case by simply setting the number of outsourced requests to only one primary cloud as non-zero.

4.2.1.3 Processing Time, Delay and Coalition Value Function

Due to cloud virtualization technology, a request actually runs on a *virtual machine* (VM) instead of a physical machine. For example, c resources can host r VMs, while each VM takes up $\frac{c}{r}$ resources. By default, we assume that each request runs on one VM¹, and if one VM takes up a unit resource, its processing time is a unit period. As a result, when r requests run on c resources, the *processing time*² of each request is r/c . As is noticed, when $r > c$, it causes a “congestion” effect and delay of processing. According to the service level agreement [130], a cloud provider bears a *penalty* if the processing time exceeds a “deadline”. By default, we set the deadline of each request as a unit time period³. For primary cloud i , given the total number of requests r_i^p and capacity c_i^p , its *delay* of processing time is

$$\tau_i(\mathbf{z}) = r_i^p / c_i^p - 1. \quad (4.4)$$

Similarly, the delay function of a coalition C_k is denoted as

$$\xi_k(\mathbf{z}) = \frac{r(C_k) - \sum_{i \in \mathcal{I}} z_{ki}}{c(C_k)} - 1, \quad (4.5)$$

where $r(C_k) - \sum_{i \in \mathcal{I}} z_{ki}$ is the number of requests processed in-house. Note that when capacity is larger than the total number of requests, the delay is negative, as the requests are processed before the deadline.

We denote the penalty function of a primary cloud i as $f_i^p(\tau_i)$. For ease of analysis, we assume all secondary clouds have a same penalty function f^s . Therefore, the penalty of coalition C_k is $f^s(\xi_k)$. However, our formulation can be extended to different penalty

¹Different request sizes can be treated as a large request being divided into smaller unit ones.

²We refer to [29] for a detailed discussion of processing time. Note that we do not consider requests being dropped by cloud providers. In practice, the rate of dropping a request is very low – for example, according to the service level agreement of Amazon EC2 [128] and Microsoft Azure [129], the rate of dropping a request is less than 0.05%, which is negligible.

³Requests with longer deadlines can be treated as several separate “virtual” requests spanning over multiple time periods. For example, if a request with a deadline of 5 unit time is received at a time t , we treat it as 5 separate virtual requests received in 5 consecutive time periods from t to $t + 4$. For each virtual request, it has a deadline of one unit time. While in the current time t , only the first among the 5 virtual requests is taken into account, and the rest 4 requests are counted as requests in future time periods.

rates¹. Accordingly, we define the value function $v(C_k; \pi; \mathbf{z})$ of coalition $C_k \in \pi$ with request strategy $\mathbf{z}$ as:

$$\begin{aligned} v(C_k; \pi; \mathbf{z}) = & \sum_{j \in C_k} p_j^s r_j^s - \sum_{i \in \mathcal{I}} p_i^p z_{ki} - \left(\sum_{j \in C_k} r_j^s \right. \\ & \left. - \sum_{i \in \mathcal{I}} z_{ki} f^s(\xi_k(\mathbf{z})) - \sum_{i \in \mathcal{I}} z_{ki} (f^s(\tau_i(\mathbf{z})) - f_i^p(\tau_i(\mathbf{z}))) \right), \end{aligned} \quad (4.6)$$

where the first term on the right hand side is the utility of processing requests from the end users. The second term is the payment to the primary clouds. The last two terms indicate the delay penalty for the requests processed in-house and in primary clouds, respectively. Note that for the case of negative delay, i.e., the request is finished before the deadline, the penalty can be viewed as a “bonus”.

4.2.2 Coalition Formation and Workload Factoring of Secondary Clouds

In this subsection, we analyze the interrelated workload factoring and coalition formation among the secondary clouds. As an example, we look at the two horizontal cloud federations in Fig. 1.1. First, the workload factoring among different horizontal federations is interdependent. For example, if Federation 1 reduces its outsourced request to the primary cloud, Federation 2 will have an incentive to increase its outsourced requests, since the processing time in the primary cloud is shorter². Second, the workload factoring and the formation of coalition structures are interdependent. For example, if the two clouds in Federation 2 decide not to cooperate, they may need more resources and the total number of outsourced requests will increase, which further impacts the outsourcing strategy of Federation 1. On the other hand, the outsourcing strategy of Federation 1 also influences whether the rest of the clouds will cooperate. For example, if Federation 1 submits fewer requests to the primary cloud, the processing time in the

¹When different secondary clouds have different penalty functions, they will have an incentive to allocate more requests to the clouds with smaller penalties. We can extend our formulation by incorporating a resource allocation subscheme among the clouds [35, 36].

²Throughout this chapter, we look at small and medium scale of cloud networks, where one secondary cloud’s request processing time in the primary cloud is impacted by the other secondary clouds, as is the case studied in [29].

Figure 4.1: Two steps to analyze the VHCF network.

primary cloud is shorter. In this case, the two clouds in Federation 2 will tend to submit more requests to the primary cloud, and it is less beneficial for them to cooperate.

Due to the above two complicated interrelations, we decompose the analysis into two steps, which is shown in Fig. 4.1. In the first step, we consider a fixed coalition structure and a workload factoring game. Next, we study the coalition formation among the secondary clouds and use a *least pessimistic core* to characterize the most stable outcome. Based on the analysis, we formulate the computation of the least pessimistic core as a bilevel optimization problem.

4.2.2.1 Workload Factoring Game (WFG) for a Fixed Coalition Structure

Definition 4.3. For a fixed coalition structure π , a **WFG** is a non-cooperative game, where the **players** are the different coalitions, the **strategy** is the number of requests outsourced to primary clouds, and the **utility** of a player is defined in Eq.(4.6).

For non-cooperative games, Nash equilibrium (NE) is a natural solution concept, where each player is not willing to unilaterally change its strategy. Given coalition structure

π , the NE strategy $\mathbf{z}_k^N$ of C_k is:

$$\mathbf{z}_k^N \in \arg \max_{\mathbf{z}_k \in \mathcal{A}_k} v(C_k; \pi; \mathbf{z}_k, \mathbf{z}_{-k}^N), \forall C_k \in \pi, \quad (4.7)$$

where $\mathbf{z}_{-k}^N$ denotes the equilibrium strategies of all the other coalitions. $\mathcal{A}_k$ is the feasible region of $\mathbf{z}_k$:

$$z_{ki} \geq 0, \forall i \in \mathcal{I}, \quad (4.8)$$

$$\sum_{i \in \mathcal{I}} z_{ki} \leq r(C_k). \quad (4.9)$$

The entire feasible solution space is $\mathcal{A} = \times_{k=1}^m \mathcal{A}_k$, where $\times$ is a Cartesian product. Eq.(4.8) means that the number of requests from coalition C_k to each primary cloud i is non-negative, while Eq.(4.9) requires that the sum of requests outsourced by each coalition C_k cannot exceed its total number of requests¹.

4.2.2.2 Coalition Formation Game with Externalities (CFG-e)

So far, we have addressed the second question in Chapter 1, i.e., an optimal workload factoring strategy in a competitive setting is an NE strategy. After we obtain the workload factoring strategy for each coalition structure with Eq.(4.7), each coalition's utility can be calculated by Eq.(4.6). To simplify notation, we omit $\mathbf{z}^N$, and denote the NE value of a coalition $C_k \in \pi$ as $v(C_k; \pi)$. As is noticed, the coalition value depends on the specific coalition structure. In the following, we answer the first question in Chapter 1 by proposing a coalition game model.

Definition 4.4. CFG-e is a coalition game played by secondary clouds, where coalition value depends on the specific coalition structure, i.e., it has **externality** effect. The **outcome** of a coalition game is a tuple $(\pi, \mathbf{x})$, where $\mathbf{x} = \langle x_1, \dots, x_j, \dots, x_{|\mathcal{J}|} \rangle$ denotes the payoff of each secondary cloud.

Since the secondary clouds are self-interested, we need to consider a *stable* outcome, i.e., no subset of players can get a higher payoff by deviating from the current outcome

¹We leave the description of solution algorithms of the WFG and the following CFG-e in Section 4.2.3.

and forming a new coalition. However, for a CFG-e, when the players want to deviate, they have to consider the response of the *residual players* (i.e., the rest of the players). There has been extensive discussion on the stability in a CFG-e. We refer to Koczy et al. [131] for a review. In this work, we use the commonly adopted *pessimistic core*¹ as the solution concept, which means that if a group of players want to deviate, they are pessimistic about their potential coalition value [133].

Definition 4.5. The **pessimistic value** of a coalition C in a CFG-e is the lowest possible value in all coalition structures Π_C which contain C : $\tilde{v}(C) = \min_{\pi \in \Pi_C} v(C, \pi)$.²

Consider an outcome $(\pi, \mathbf{x})$, where $x_j \geq 0, \forall j \in \mathcal{J}$, it is in the *pessimistic core* if $\sum_{j \in C} x_j \geq \tilde{v}(C), \forall C \in \mathcal{C}$, and $\sum_{j \in C} x_j = v(C, \pi), \forall C \in \pi$. The two constraints ensure that 1) the sum of payoff for each subset of players is no smaller than the pessimistic value of their intended coalition and 2) the sum of payoff of all the coalition members is equal to the coalition value. However, the pessimistic core can be empty, or there can be multiple cores. Thus we define a “least pessimistic core” to denote the set of most stable outcomes for a CFG-e.

Definition 4.6. The **least pessimistic core** of a CFG-e is the outcome $(\pi, \mathbf{x})$ with the smallest ε satisfying

$$\sum_{j \in C} x_j \geq \min_{\pi \in \Pi_C} v(C, \pi) - \varepsilon, \forall C \in \mathcal{C} \quad (4.10)$$

$$\sum_{j \in C} x_j = v(C, \pi), \forall C \in \pi \quad (4.11)$$

$$x_j \geq 0, \forall j \in \mathcal{J} \quad (4.12)$$

When $\varepsilon > 0$, a set of players prefer to deviate when the loss of deviation is no larger than ε , while $\varepsilon \leq 0$ means that they prefer to stay at the current outcome even if they can get a reward of $-\varepsilon$ if they deviate. Therefore, the outcome with the smallest ε is the most stable. Note that in Eq.(4.10), to obtain the pessimistic value of a coalition C , we need to go through all the coalition structures that contain C , which is impractical when

¹We use core instead of Shapley value [132] because to incentivize the self-interested clouds, stability is more desired than fairness.

²Our framework can be easily extended to the optimistic core [131] by defining an optimistic value $\hat{v}(C) = \max_{\pi \in \Pi_C} v(C, \pi)$.

the scale of the problem is large. Before we finally formulate the bilevel optimization problem to obtain the least pessimistic core and the corresponding outcome, we simplify Eq.(4.10) with two important properties of the proposed CFG-e, namely, *superadditivity* and *positive externality*.

Lemma 4.7. *When cooperation incurs no cost, the proposed CFG-e is **superadditive**, i.e., for any two coalitions $C_k, C_{k'}$, there is $v(C_k \cup C_{k'}; \{C_k \cup C_{k'}\} \cup \pi_r) \geq v(C_k; \{C_k, C_{k'}\} \cup \pi_r) + v(C_{k'}; \{C_k, C_{k'}\} \cup \pi_r)$, where π_r denotes a partition of the residuals $\mathcal{J} \setminus (C_k \cup C_{k'})$. Coalition structure $\{C_k \cup C_{k'}\} \cup \pi_r$ means that C_k and $C_{k'}$ cooperate, and $\{C_k, C_{k'}\} \cup \pi_r$ means that they do not cooperate.*

Proof. The proof is in two phases. First, we prove that for two coalitions C_k and $C_{k'}$, if the total number of their outsourced requests after cooperation equals the sum of outsourced requests before cooperation, their total delay is no larger after cooperation. Second, we prove that, if the total number outsourced requests can be changed, the total delay (utility) is still no larger (smaller) after cooperation.

1) *Total number of outsourced requests do not change.* We first assume that after C_k and $C_{k'}$ form a new coalition, the total number of outsourced requests equals the sum of their outsourced requests before cooperation. In this case, only the delay penalty (the third term in Eq.(4.6)) in the coalition value function is changed. Denote the number of requests processed in-house as $r_{in}(C_k) = r(C_k) - \sum_{i \in \mathcal{I}} z_{ki}$ and $r_{in}(C_{k'}) = r(C_{k'}) - \sum_{i \in \mathcal{I}} z_{k'i}$, we can obtain the total delay of C_k and $C_{k'}$ under two cases: 1) C_k and $C_{k'}$ cooperate and form a new coalition; 2) C_k and $C_{k'}$ do not cooperate. By subtracting the total delay of these two cases, we have:

$$\begin{aligned} & r_{in}(C_k) \left(\frac{r_{in}(C_k)}{c(C_k)} - 1 \right) + r_{in}(C_{k'}) \left(\frac{r_{in}(C_{k'})}{c(C_{k'})} - 1 \right) - (r_{in}(C_k) + r_{in}(C_{k'})) \\ & \left(\frac{r_{in}(C_k) + r_{in}(C_{k'})}{c(C_k) + c(C_{k'})} - 1 \right) = \frac{r_{in}^2(C_k)}{c(C_k)} + \frac{r_{in}^2(C_{k'})}{c(C_{k'})} - \frac{(r_{in}(C_k) + r_{in}(C_{k'}))^2}{c(C_k) + c(C_{k'})} \\ & = \frac{(r_{in}(C_k)c(C_{k'}) - r_{in}(C_{k'})c(C_k))^2}{c(C_k)c(C_{k'})(c(C_k) + c(C_{k'}))} \geq 0. \end{aligned}$$

The equality holds (i.e., the total delays of the above two cases are equal) when the processing rates of coalitions C_k and $C_{k'}$ are the same (i.e., $\frac{r_{in}(C_k)}{c(C_k)} = \frac{r_{in}(C_{k'})}{c(C_{k'})}$). Therefore, the total delay (also for unit delay, since the total number of requests is the same

whether the two coalitions cooperate or not) is always no larger when clouds cooperate than the total delay when they do not cooperate.

2) *Total number of outsourced requests can be changed.* If the new coalition $C_k \cup C_{k'}$ can change the number of outsourced requests, it can at least maintain the current request strategy (or change it only when it is beneficial). As a result, the total delay of the merged new coalition is still always no larger than the total delay of the two separate coalitions. Correspondingly, the utility of the merged new coalition is always no smaller than the sum of utility of C_k and $C_{k'}$ if these two coalitions do not cooperate. $\square$

Lemma 4.8. *When cooperation incurs no cost, the proposed CFG-e has **positive externality**, i.e., for any coalitions $C_k, C_{k'}, C_{k''} \subseteq \mathcal{J}$, and a partition π_r of the residuals $\mathcal{J} \setminus (C_k \cup C_{k'} \cup C_{k''})$, the value of C_k is always no smaller when $C_{k'}$ and $C_{k''}$ cooperate: $v(C_k; \{C_k, C_{k'} \cup C_{k''}\} \cup \pi_r) \geq v(C_k; \{C_k, C_{k'}, C_{k''}\} \cup \pi_r)$.*

Proof sketch. At the NE strategy of a coalition, if the coalition increases its outsourced requests by a very small amount (it brings no change of the processing time in either the coalition or the primary clouds), the benefit from the decrease of delay penalty in-house equals the payment to the primary clouds plus the delay penalty from the primary clouds. According to the proof of Lemma 4.7, when the new coalition $C_k \cup C_{k'}$ maintains its request strategy, its unit delay never increases. Thus the new coalition has an incentive to decrease its outsourced request so that the delay in-house increases, the delay in primary clouds decreases, and the equilibrium is reached again. When the delay in primary clouds decreases, the delay penalty of other coalitions decreases, and their utility increases. $\square$

Theorem 4.9. *When cooperation incurs no cost, the grand coalition is the most stable coalition structure.*

Proof sketch. When cooperation incurs no cost, superadditivity implies that cooperation is beneficial to the coalition members, while positive externality implies that cooperation is beneficial to the clouds outside the coalition. Thus the grand coalition

will always be the most stable coalition structure. $\square$

However, forming coalitions incurs additional cost, such as extra hypervisor machines, administration cost and data transmission cost. Therefore, we add a cost term to the coalition value function:

$$v'(C_k, \pi) = v(C_k, \pi) - C^{co}(C_k). \quad (4.13)$$

Since the coalition cost is generated in the intra communication within a coalition, we assume it depends only on the coalition itself, i.e., it does not have externality effect.

Theorem 4.10. *When coalition cost has no externality effect, the proposed CFG-e has positive externalities.*

Proof sketch. This is a direct extension of Lemma 4.8. $\square$

According to Theorem 4.10, the pessimistic value of a coalition C is associated with the coalition structure π where all residual players form *singleton coalitions*, i.e., each coalition has only one member. With this property, the pessimistic value of all coalitions can be obtained directly, which greatly simplifies Eq.(4.10). Combining this property with Eqs.(4.7), (4.10)-(4.12), the solution (i.e., the most stable outcome) of this CFG-e is expressed with the following bilevel optimization problem:

$$\min_{\pi, \mathbf{x}, \mathbf{z}^N} \quad \varepsilon \quad (4.14)$$

$$\text{s.t.} \quad \sum_{j \in C} x_j \geq \tilde{v}(C) - \varepsilon, \quad \forall C \in \mathcal{C} \quad (4.15)$$

$$\sum_{j \in C_k} x_j = v'(C_k; \pi; \mathbf{z}^N), \quad \forall C_k \in \pi \quad (4.16)$$

$$x_j \geq 0, \quad \forall j \in \mathcal{J} \quad (4.17)$$

$$\pi \in \Pi \quad (4.18)$$

$$\mathbf{z}_k^N \in \operatorname{argmax}_{\mathbf{z}_k \in \mathcal{A}_k} v(C_k; \pi; \mathbf{z}_k, \mathbf{z}_{-k}^N), \quad \forall C_k \in \pi \quad (4.19)$$

Eqs.(4.14)-(4.18) describe the upper level problem of computing the pessimistic core, while Eq.(4.19) denotes the lower level problem of computing an NE strategy for a fixed coalition structure π . Eqs.(4.14)-(4.17) compute the least ε value for a fixed π . Eq.(4.18) denotes the feasible coalition structure space, which will be introduced in detail in Section 4.2.3.

4.2.3 Computing Least Pessimistic Core

Due to the two-level structure and involved form of coalition value functions, there is no existing method to solve the above optimization problem directly. In this subsection, we decompose the bilevel problem and design algorithms for both levels. In the lower level, we propose two algorithms CLONE and CLOSE. CLONE employs the constraint generation method, and is a generic algorithm. CLOSE is a more efficient algorithm which employs several structural properties of the lower level problem, but is limited to concave coalition value functions. In the upper level, we propose Brute-Force and CUBE. CUBE implements a binary search method with two heuristic rules and is more efficient than Brute-Force.

4.2.3.1 Solving the Lower Level WFG

Given a coalition structure π , the lower level problem in Eq.(4.19) is a feasibility problem:

$$\max_{\mathbf{z}^N} 0 \quad (4.20)$$

$$v(C_k; \pi; \mathbf{z}^N) \geq v(C_k; \pi; \mathbf{z}_k, \mathbf{z}_{-k}^N), \forall \mathbf{z}_k \in \mathcal{A}_k, C_k \in \pi \quad (4.21)$$

$$\mathbf{z}^N \in \mathcal{A} \quad (4.22)$$

The General Algorithm CLONE: Unfortunately, the number of constraints in Eq.(4.21) is infinite since $\mathbf{z}_k \in \mathcal{A}_k$ is a continuous parameter. We use the constraint generation method [134] to deal with the infinite constraint space. The basic idea is to start with solving a relaxation of the original problem by sampling a small subset

of constraints and then use a sub-routine to find violated constraints, and gradually expand the set by adding the violated constraints to the relaxed problem until no violated constraint is found.

Algorithm 5: CLONE

```

1 Sample a subset  $\Omega$  of constraints from  $\mathcal{A}$ ;
2 repeat
3    $\mathbf{z}^N \leftarrow$  solution of master problem;
4    $\mathbf{z}_k^{opt} \leftarrow$  solution of slave problem,
5    $Df_k^{opt} \leftarrow$  objective of slave problem;
6   for  $C_k \in \pi$  do
7     if  $Df_k^{opt} > e_k$  then add  $\mathbf{z}_k^{opt}$  to  $\Omega$ ;
8 until  $Df_k^{opt} \leq e_k, \forall C_k \in \pi$ ;
9 return  $\mathbf{z}_k^N, v(C_k; \pi; \mathbf{z}^N), \forall C_k \in \pi$ .
```

CLONE (Compute Lower level Optimization with coNstraint gEneration), as shown in Algorithm 5, begins by randomly sampling a finite subset Ω_k of constraints from $\mathcal{A}_k$ for each coalition C_k . A relaxed *master* problem is solved which is in the same form as the original feasibility problem in Eqs.(4.20)-(4.22) and returns the feasible solution $\mathbf{z}^*$. The only difference is that the master problem replaces $\mathcal{A}_k$ with Ω_k in Eq.(4.21). Constraint generation is then applied to find the “mostly violated” constraint for each coalition C_k , as is shown in the following *slave* problem:

$$\max_{\mathbf{z}_k \in \mathcal{A}_k} Df_k = v(C_k; \pi; \mathbf{z}_k, \mathbf{z}_{-k}^*) - v(C_k; \pi; \mathbf{z}_k^*, \mathbf{z}_{-k}^*). \quad (4.23)$$

The most violated constraint refers to the parameter $\mathbf{z}_k$ that maximizes the difference between the utility of coalition C_k computed by $\mathbf{z}_k$ and by the optimal solution $\mathbf{z}_k^*$ of the master problem. The solutions of the slave problem return the set of mostly violated constraints, which will be added to the set of constraints Ω of the master problem. The process iterates until convergence, i.e., the optimal value Df_k^{opt} of the slave problem corresponding to each coalition $C_k \in \pi$ is no larger than a small threshold e_k . This threshold is set as the multiplication of the coalition’s gross revenue $\sum_{j \in C_k} r_j^s p_j^s$ and a small ratio e (we will evaluate the runtime of the algorithm with different scales of e in Section 4.2.4). In this way, we decompose the original optimization problem into a

series of master-slave non-linear optimization problems, which have a finite number of constraints and can be solved directly by existing optimization solvers like KNITRO.

The Faster Algorithm CLOSE: With CLONE, we can solve the lower level optimization with any form of coalition value functions. However, it solves a series of non-linear master-slave problems, and thus becomes rather slow for large scale settings¹. In the following, we first demonstrate that under the commonly employed linear delay penalty rules [135–137] (i.e., $f_i^p(\tau_i) = \beta_i^p \tau_i$, $f^s(\xi_k) = \beta^s \xi_k$, where β_i^p and β^s are penalty rates for primary cloud i and secondary clouds, resp.), the value function of each coalition C_k is strictly concave w.r.t. its request strategy vector $\mathbf{z}_k$. We then use this property to establish the existence and uniqueness of the NE. Based on the theoretical analysis, we introduce a more efficient algorithm to solve the lower level problem.

Algorithm 6: CLOSE

```

1  $\mathbf{z}^* \leftarrow$  solution of Eq.(4.24);
2 for  $C_k \in \pi$  do
3   for  $i \in \mathcal{I}$  do
4     if  $z_{ki}^* < 0$  then
5       Replace the corresponding equation in Eq.(4.24) by  $z_{ki} = 0$ ;
6  $\mathbf{z}^N \leftarrow$  Solution of the new equation system;
7 return  $\mathbf{z}_k^N, v(C_k; \pi; \mathbf{z}^N), \forall C_k \in \pi$ .
```

Lemma 4.11. *With linear penalty functions $f_i^p(\tau_i) = \beta_i^p \tau_i$, $f^s(\xi_k) = \beta^s \xi_k$, the coalition value function defined in Eq.(4.6) is strictly concave w.r.t. its request strategy $\mathbf{z}_k$.*

Proof. Let $A_k = \frac{\beta^s}{c(C_k)}$, $D_k = \sum_{j \in C_k} p_j^s r_j^s - \beta^s r(C_k) (\frac{r(C_k)}{c(C_k)} - 1)$, $v_k = v(C_k; \pi; \mathbf{z})$, we rewrite the value function in Eq.(4.6) as: $v_k = -A_k (\sum_{i \in \mathcal{I}} z_{ki})^2 - \sum_{i \in \mathcal{I}} (\frac{(\beta^s - \beta_i^p) z_{ki}}{c_i^p} \sum_{C_{k'} \in \pi} z_{k'i}) + (\frac{2\beta^s r(C_k)}{c(C_k)} - \beta_i^p) \sum_{i \in \mathcal{I}} z_{ki} + D_k - \sum_{i \in \mathcal{I}} p_i^p z_{ki}$. The Hessian matrix of $-v_k$ is defined as $H_{ll'}(-v_k) = \frac{\partial^2 v_k}{\partial z_{kl} \partial z_{kl'}}$, $l, l' \in \mathcal{I}$. Substitute v_k with the above equation, we have:

$$H_{ll'}(-v_k) = \begin{cases} 2(A_k + \frac{(\beta^s - \beta_i^p) p_i^p}{c_i^p}), & l = l', \\ 2A_k, & l \neq l'. \end{cases} \quad \text{Let } B_l = \frac{\beta^s - \beta_i^p}{c_i^p}. \text{ For any upper left } \tilde{l} \times \tilde{l}$$

¹CLONE is still meaningful because when coalitions' value functions are not strictly concave, only CLONE can be applied.

matrix, where $\tilde{l} = 1, \dots, i, \dots, |\mathcal{I}|$, its determinant

$$\begin{aligned}
 \det H_{\tilde{l}} &= 2^{\tilde{l}} \begin{vmatrix} A_k + B_1 & A_k & \cdots & A_k \\ A_k & A_k + B_2 & \cdots & A_k \\ \cdots & \cdots & \cdots & \cdots \\ A_k & A_k & \cdots & A_k + B_{\tilde{l}} \end{vmatrix} = 2^{\tilde{l}} \begin{vmatrix} A_k + B_1 & A_k & \cdots & \cdots & A_k \\ -B_1 & B_2 & 0 & \cdots & 0 \\ \cdots & 0 & B_3 & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & 0 \\ -B_1 & 0 & \cdots & 0 & B_{\tilde{l}} \end{vmatrix} \\
 &= 2^{\tilde{l}} \begin{vmatrix} A_k + B_1 + A_k \sum_{l=2}^{\tilde{l}} \frac{B_1}{B_l} & A_k & \cdots & \cdots & A_k \\ 0 & B_2 & 0 & \cdots & 0 \\ \cdots & 0 & B_3 & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & 0 & \cdots & 0 & B_{\tilde{l}} \end{vmatrix} = (A_k + B_1 + A_k \sum_{l=2}^{\tilde{l}} B_1/B_l) \Pi_{l=2}^{\tilde{l}} B_l,
 \end{aligned}$$

Since $A_k > 0, B_l > 0$, then $\det H_{\tilde{l}} > 0, \forall \tilde{l} \in \mathcal{I}$, i.e., the Hessian matrix of $-v_k$ w.r.t. $\mathbf{z}_k$ is positive definite everywhere. Thus $-v_k$ is strictly convex w.r.t. $\mathbf{z}_k$, while $v(C_k; \pi; \mathbf{z}_k)$ is strictly concave. $\square$

The strict concavity of coalition function is intuitive: when the number of outsourced requests is small, the coalition value increases with the outsourced requests because the in-house overload is relieved. When the number of outsourced requests is too large, the coalition value begins to decrease because the benefit of relieving in-house overload is smaller than the cost of outsourcing.

Theorem 4.12. (Existence and uniqueness of equilibrium) *With linear delay penalty, the pure strategy Nash equilibrium for the WFG given a coalition structure π is non-empty and unique.*

Proof. For each coalition C_k as a player of the WFG, given the strategies $\mathbf{z}_{-k}$ of the other players, it will select the optimal strategy $\mathbf{z}_k$ that maximizes its value, which is a mapping $\phi_k : \mathcal{A}_{-k} \rightarrow \mathcal{A}_k$ defined as $\mathbf{z}_k = \phi_k(\mathbf{z}_{-k})$. According to Lemma 4.11, the value function $v(C_k; \pi; \mathbf{z})$ of each coalition C_k is strictly concave w.r.t. $\mathbf{z}_k$. Combined with the compactness (closed and bounded) of both the domain $\mathcal{A}_{-k}$ and the range $\mathcal{A}_k$, we have that the mapping $\phi_k(\mathbf{z}_{-k})$ is non-empty and unique. Also, the compactness

of $\mathcal{A}_{-k}$ implies that the mapping $\phi_k : \mathcal{A}_{-k} \rightarrow \mathcal{A}_k$ is upper hemicontinuous. The non-emptiness, uniqueness (point to point), and upper hemicontinuity are naturally inherited by the mapping $\phi : \mathcal{A} \rightarrow \mathcal{A}$ defined as $\phi(\mathbf{z}) = \times_k^m \phi_k(\mathbf{z}_{-k})$. Moreover, since the vertices of $\mathcal{A}$ defined in Eqs.(4.8)-(4.9) are affinely independent, $\mathcal{A}$ is a simplex (and closed). Applying Kakutani's fixed point theorem [138], there exists a unique $\mathbf{z}^N \in \mathcal{A}$ such that $\mathbf{z} = \phi(\mathbf{z})$. Thus, the pure strategy NE of our problem is non-empty and unique. $\square$

Based on Theorem 3, we design a more efficient algorithm for the lower level problem.

Theorem 4.13. *When the feasible region $\mathcal{A}$ in Eq.(4.22) is relaxed to $\mathbb{R}^{m \times n}$, the lower level optimization problem is equivalent to solving the following linear equation system:*

$$\frac{\partial v(C_k; \pi; \mathbf{z})}{\partial z_{ki}} = 0, \quad \forall i \in \mathcal{I}, \quad \forall C_k \in \pi. \quad (4.24)$$

Proof. When the feasible region is relaxed to $\mathbf{z} \in \mathbb{R}^{m \times n}$, then the computation of the NE for a fixed coalition structure π is equivalent to: $\max_{\mathbf{z}_k} v(C_k; \pi; \mathbf{z}), \forall C_k \in \pi$. According to Lemma 4.11, when the request strategy $\mathbf{z}_{-k}$ of the other coalitions is fixed, the value function of each coalition is strictly concave w.r.t. $\mathbf{z}_k$. Employing the first order condition, for coalition C_k , its unique optimal strategy vector can be computed as: $\frac{\partial v_k}{\partial z_{ki}} = 0, \forall i \in \mathcal{I}$. Considering all the coalitions, we obtain Eq.(4.24). $\square$

Unfortunately, the solution of Eq.(4.24) may not always lie in the feasible region in Eqs.(4.8)-(4.9). In practice, the price of primary clouds is much lower than that of secondary clouds. Thus, we always assume their penalty rate is also lower than that of secondary clouds, i.e., $\beta^s > \beta_i^p$.

Theorem 4.14. *The solution of Eq.(4.24) is guaranteed to satisfy the feasibility constraints in Eq.(4.9).*

Proof. With Theorem 4.13, for each coalition $C_k \in \pi$, we compute its NE strategy vector with $\frac{2\beta^s}{c(C_k)}(r(C_k) - \sum_{i \in \mathcal{I}} z_{ki}) - \frac{\beta^s - \beta_i^p}{c_i^p}(\sum_{C_{k'} \in \pi} z_{k'i} + z_{ki}) - p_i^p - \beta_i^p = 0$. Extract $r(C_k) - \sum_{i \in \mathcal{I}} z_{ki}$, we have $r(C_k) - \sum_{i \in \mathcal{I}} z_{ki} = \frac{c(C_k)}{2\beta^s}(\frac{\beta^s - \beta_i^p}{c_i^p}(\sum_{C_{k'} \in \pi} z_{k'i} + z_{ki}) + p_i^p)$. Due to the assumption above, we have $\beta^s > \beta_i^p$. Moreover, we have $z_{ki} \geq 0$, and thus $\sum_{i \in \mathcal{I}} z_{ki} \leq r(C_k)$ is satisfied. $\square$

Figure 4.2: Two cases of NE. x and y axes denote the request strategy of coalitions C_1 and C_2 , resp. The two lines L_1 and L_2 denote the *best response curves* of coalitions C_1 and C_2 when the feasible region $\mathcal{A}$ is relaxed, while the dashed line segments are outside the feasible region. Following Theorem 4.15, they are replaced by the dotted line segments.

Theorem 4.14 implies that the feasibility constraint in Eq.(4.9) can be omitted with the assumption that $\beta^s > \beta_i^p$. In the next, we further discuss the feasibility constraint in Eq.(4.8). As we know, each partial differential equation in Eq.(4.24) can be viewed as a *best response curve* of a coalition given the strategies of other coalitions (Fig. 4.2). The intersection of these curves is the solution of Eq.(4.24). However, for each curve, there is a segment that lies outside the feasible region (the dashed line segments in Fig. 4.2).

Theorem 4.15. *For the best response curves of a coalition described in Eq.(4.24), the segments that lie outside the feasible region (i.e., $z_{ki} < 0$) should be replaced by the curve $z_{ki} = 0$.*

Proof. According to the definition of strict concavity, for the best response curve where $z_{ki} < 0$, we have $v(C_k; \pi; z''_{ki} = 0, \mathbf{z}_{k,-i}, \mathbf{z}_{-k}) > \lambda v(C_k; \pi; z_{ki} < 0, \mathbf{z}_{k,-i}, \mathbf{z}_{-k}) + (1 - \lambda)v(C_k; \pi; z'_{ki} > 0, \mathbf{z}_{k,-i}, \mathbf{z}_{-k})$, where $\mathbf{z}_{k,-i}$ is the request strategy to primary clouds except i , z_{ki} is on the best response curve, $z'_{ki} > 0$ is any value in the feasible region $\mathcal{A}$. Since $z''_{ki} = 0$ is between z_{ki} and z'_{ki} , then $0 < \lambda < 1$. Also we have $v(C_k; \pi; z''_{ki} = 0, \mathbf{z}_{k,-i}, \mathbf{z}_{-k}) < v(C_k; \pi; z_{ki} < 0, \mathbf{z}_{k,-i}, \mathbf{z}_{-k})$. Therefore, $v(C_k; \pi; z''_{ki} = 0, \mathbf{z}_{k,-i}, \mathbf{z}_{-k}) > v(C_k; \pi; z'_{ki} > 0, \mathbf{z}_{k,-i}, \mathbf{z}_{-k})$ holds for any $z'_{ki} > 0$ and the best response curve should be $z_{ki} = 0$. $\square$

Based on the properties above, we propose CLOSE (Compute Lower level Optimization with Strictly concavE value function) to solve the lower level problem in Eqs.(4.20)-(4.22), as shown in Algorithm 6.

4.2.3.2 Solving the Upper Level CFG-e

Since the coalition value function is in an implicit form which depends on the solution of the lower level problem, the upper level problem cannot be formulated as a single integer-linear program. Instead, a separate linear program (Eqs.(4.14)-(4.17)) has to be solved to find the least core of each coalition structure. A natural idea is to do a brute-force search within Π to find the least pessimistic core.

Algorithm 7: Brute-Force

```

1  $\varepsilon^* \leftarrow M$ ;
2 for  $\Pi_K, K = 1 : |\mathcal{J}|$  do
3    $\varepsilon_K^* \leftarrow M$ ;
4   for node  $\pi \in \Pi_K$  do
5     Solve Eq.(4.19) for  $\pi$ , get coalition values;
6      $\varepsilon_\pi \leftarrow$  solution of Eqs.(4.14)-(4.17);
7     if  $\varepsilon_K^* > \varepsilon_\pi$  then  $\varepsilon_K^* \leftarrow \varepsilon_\pi$ ;
8   if  $\varepsilon^* > \varepsilon_K^*$  then  $\varepsilon^* \leftarrow \varepsilon_K^*$ ;
9 return  $\varepsilon^*$ 

```

The Exact Algorithm Brute-Force: We first introduce a *coalition structure graph*.

Definition 4.16. A **coalition structure graph (CSG)** Π is a graph with $|\mathcal{J}|$ levels, where each node denotes a coalition structure, and each **level** $\Pi_K, K = 1, 2, \dots, |\mathcal{J}|$ contains the coalition structures that have K coalitions. Fig. 4.3 shows an example CSG with 3 secondary clouds.

A brute-force searching algorithm, as shown in Algorithm 7, traverses all the nodes from bottom-up, level by level. For each node, a linear program (Eqs.(4.14)-(4.17)) is called as a subroutine to obtain the least core value ε_π for the node. ε^* and ε_K^* respectively denote the least core value for Π and in the level Π_K , which are initialized as a very large value M .

The Heuristic Algorithm CUBE: Note that the number of coalition structures is a *Bell number* w.r.t. the number of secondary clouds n , which belongs to one of the most notorious large numbers in the combinatorial mathematics and is of order $O(n^n)$ and $\omega(n^{n/2})$ [139]. As a result, it is rather inefficient to do a brute-force search within the exponentially large solution space. In fact, finding the least core for large scale coalition

Figure 4.3: A CSG with 3 secondary clouds. The bottom node (level 1) is the grand coalition, the top node (level 3) has 3 singleton coalitions, and all nodes in level 2 have 2 coalitions.

games with externalities, which belongs to hardest class of *combinatorial optimization* problems, has long been an open problem. Although there have been attempts to solve much simpler problems [140–142], no attempts have been made to obtain the least core for large scale coalition games with externalities. In this work, we design a binary search algorithm, CUBE, to solve the upper level problem, which implements two heuristic pruning rules based on the properties of the CFG-e.

Heuristic Rule 1: When $\varepsilon_K^* < \varepsilon_{K'}^*$, search in levels closer to Π_K . $\varepsilon_K^* < \varepsilon_{K'}^*$ implies that the nodes in Π_K tend to be more stable than those in $\Pi_{K'}$. Thus we put more search effort to the levels closer to Π_K .

Heuristic Rule 2: For a coalition $C \in \pi$, if the ratio of its request and capacity $\frac{r(C)}{c(C)}$ satisfies $\frac{r(C)}{c(C)} < 1$ or $\frac{r(C)}{c(C)} > \theta \times \frac{\sum_{C \in \pi} r(C)}{\sum_{C \in \pi} c(C)}$ ¹, prune this node π . The intuition behind this heuristic rule is that when a coalition C has redundant resources ($\frac{r(C)}{c(C)} < 1$) or has too low available resources ($\frac{r(C)}{c(C)} > \theta \times \frac{\sum_{C \in \pi} r(C)}{\sum_{C \in \pi} c(C)}$), the coalition value is expected to be low and thus the coalition will not form. If any such coalition C exists, a coalition structure which contains C cannot exist either. With smaller values of θ , the pruning is more aggressive, while sacrificing optimality.

¹It reveals the average request-capacity ratio of all secondary clouds.

Based on the above two rules, we propose CUBE (Compute Upper level problem with Binary sEearch). As shown in Algorithm 8, the algorithm iteratively searches within the smallest level Π_K to the largest level $\Pi_{K'}$, which are initialized as 1 and $|\mathcal{J}|$, respectively. For $K < K'$, meaning that there are more than 2 levels of nodes, CUBE computes the least core $\varepsilon_K^*, \varepsilon_{K'}^*$ of these two levels (Lines 4-11). In Lines 7-8, it skips any node which satisfies *heuristic rule 2*. If $\varepsilon_K^* < \varepsilon_{K'}^*$, according to *heuristic rule 1*, the largest level becomes $K' = \lfloor (K + K')/2 \rfloor$, which means that half of the levels closer to level $\Pi_{K'}$ are pruned. Else, the smallest level is updated as $K \leftarrow \lceil (K + K')/2 \rceil$.

Algorithm 8: CUBE

```

1  $\varepsilon^* \leftarrow M; K \leftarrow 1, K' \leftarrow |\mathcal{J}|;$ 
2 while  $K < K'$  do
3    $\varepsilon_K^* \leftarrow M, \varepsilon_{K'}^* \leftarrow M;$ 
4   for node  $\pi \in \Pi_K$  do
5     for  $C \in \pi$  do
6       if  $\frac{r(C)}{c(C)} \geq 1$  &  $\frac{r(C)}{c(C)} \leq T_2$  then
7         Solve Eq.(4.19) for  $\pi$ , get coalition values;
8          $\varepsilon_\pi \leftarrow$  solution of Eqs.(4.14)-(4.17);
9         if  $\varepsilon_K^* > \varepsilon_\pi$  then  $\varepsilon_K^* \leftarrow \varepsilon_\pi;$ 
10  Repeat Lines 5-10 for  $\Pi_{K'}$ , get  $\varepsilon_{K'}^*$ ;
11  if  $\varepsilon_K^* < \varepsilon_{K'}^*$  then  $K' \leftarrow \lfloor (K + K')/2 \rfloor;$ 
12  else  $K \leftarrow \lceil (K + K')/2 \rceil;$ 
13 return  $\varepsilon^*;$ 

```

4.2.4 Experimental Results

In this subsection, we 1) show through a simple example the interrelated workload factoring and coalition formation, and the general process of obtaining the least pessimistic core; 2) evaluate superadditivity and externality; 3) evaluate coalition formation under different VHCF scenarios; 4) show the improvement of total utility by using the VHCF network; 5) evaluate the performance of different algorithms.

4.2.4.1 Obtaining Least Pessimistic Core: A Simple Scenario

To show the interrelated workload factoring and coalition formation, and the general process of obtaining the least pessimistic core, we first study a simple scenario of 1

primary cloud and 3 secondary clouds. Note that in this part, we do not intend to evaluate how different parameters (e.g., the price, penalty of secondary clouds, coalition cost, demand and supply sizes. We refer to Table 4.3 for a summary of parameters that are mentioned in the experimental settings.) influence the coalition formation results among the secondary clouds (which will be evaluated in Section 4.2.4.3). As a result, in this part, we just assign arbitrary values to these parameters, as described in the following.

Parameter	Description
p_i^p	price per request of primary cloud i
c_i^p	capacity of primary cloud i
β_i^p	penalty rate of primary cloud i
p_j^s	price per request of secondary cloud j
c_j^s	capacity of secondary cloud j
r_j^s	number of requests of secondary cloud j
β^s	penalty rate of secondary clouds
c^{co}	unit coalition cost
γ	coalition size impact factor

TABLE 4.3: Summary of parameters mentioned in the experimental settings. We evaluate the influence of these parameter values on coalition formation results in Section 4.2.4.3.

By default, we set the price of the primary cloud as 1. Correspondingly, we set the price vector of the 3 secondary clouds as $\langle 2, 1.98, 2.01 \rangle$ — approximately twice as that of the primary cloud's. The prices of the secondary clouds are set close because, to ensure interoperability among different secondary clouds, the application service in these cooperating clouds should be similar, while the prices of their service should also be close. The capacity of the primary cloud is set as 40. The capacity and request vectors of the 3 secondary clouds are $\langle 40, 50, 60 \rangle$ and $\langle 60, 40, 90 \rangle$, resp. Following the line of research of [135–137], throughout this subsection, we adopt the linear penalty rules. The penalty rates of the primary clouds and secondary clouds are set as $\beta^p = 1$ and $\beta^s = 2$, resp. Note that when penalty rate equals price, it means if processing delay is twice as much as the deadline, the penalty for delay equals the price paid and the provider gets no payment from the user.

Coalition cost is defined as¹: $C^{co}(C_k) = c^{co}(|C_k| - 1)^\gamma$, where c^{co} is the unit coalition

¹Each secondary cloud will estimate the cost if it joins a coalition. The detailed modelling of coalition cost is out of scope of this research.

cost. $(|C_k| - 1)^\gamma$ is determined by the coalition size $|C_k|$. For example, when a coalition is a singleton, there is no coalition cost. Whenever a new member enters the coalition, it incurs additional coalition cost. γ is a factor which measures the extent to which coalition size influences the coalition cost. Larger γ values indicate larger coalition size effect. We set $\gamma = 1$ (i.e., coalition cost is linear w.r.t. coalition size), $c^{co} = 6$ (i.e., when a new member joins the coalition, an additional cost of 6 is produced).

Coalition structure	NE request	Coalition value	Least ε	Payoff division
$\{1, 2, 3\}$	43.1	421.9	0	127.3, 119.2, 175.4
$\{1, 2\}, \{3\}$	23.1, 29.3	241.9, 181.2	1.1611	123.8, 118.1, 181.2
$\{1, 3\}, \{2\}$	47.6, 1.2	296.7, 119.3	5.90571	127.2, 119.3, 169.5
$\{1\}, \{2, 3\}$	29.4, 23.3	130.1, 292.7	0.941181	130.1, 118.2, 174.5
$\{1\}, \{2\}, \{3\}$	28.5, 0, 28.0	125.0, 119.2, 175.4	2.3508	125.0, 119.2, 175.4

TABLE 4.4: NE, coalition value and least pessimistic core (the outcome highlighted in yellow).

Coalition	$\{1\}$	$\{2\}$	$\{3\}$	$\{1, 2\}$	$\{1, 3\}$	$\{2, 3\}$	$\{1, 2, 3\}$
Pessimistic value	125.0	119.2	175.4	241.9	296.7	292.7	421.9

TABLE 4.5: Pessimistic value

First, we call CLONE/CLOSE to compute the NE strategies and coalition values of different coalition structures, which are shown in the 2nd and 3rd columns of Table 4.4. We can see that the workload factoring and the coalition formation are interdependent. For example, the request strategy of coalition $\{1\}$ is 29.4 when secondary clouds 2 and 3 cooperate, while 28.5 when they do not cooperate. Alternatively, it can be viewed as the request strategy of $\{1\}$ influences whether secondary clouds 2 and 3 cooperate. We can also see that the coalition value of a coalition depends on the coalition structure, i.e., the externality effect, which will be studied in detail in the next part.

After we obtain the coalition values for each coalition structure, we can get the pessimistic value of each coalition. For example, the two coalition structures which contain coalition $\{1\}$ are $\{1\}, \{2, 3\}$ and $\{1\}, \{2\}, \{3\}$. According to Definition 4.5, the pessimistic value of $\{1\}$ is the smaller one among the two values 130.1 and 125.0. Therefore, the pessimistic value of coalition $\{1\}$ is $\tilde{v}(\{1\}) = 125.0$. In fact, this is the direct result of Theorem 4.10, since in coalition structure $\{1\}, \{2\}, \{3\}$, both $\{2\}$ and

$\{3\}$ are singleton coalitions. Similarly, we can obtain the pessimistic value of all the coalitions in Table 4.5.

Last, for each coalition structure, we call the linear program in Eqs.(4.14)-(4.17) to compute the least ε and the corresponding payoff of each secondary cloud, as is shown in the last two columns of Table 4.4. We can see that only the least ε of the grand coalition is 0, which means that only the grand coalition and its corresponding payoff division is in the least pessimistic core (highlighted in yellow). Take another coalition structure $\{1\}, \{2, 3\}$ as an example, the payoff of secondary cloud 2 is 118.2, while the pessimistic value of coalition $\{2\}$ is $119.2 > 118.2$. Therefore, secondary cloud 2 has an incentive to deviate from the current outcome and form a singleton coalition. Interestingly, although secondary cloud 1 gets the highest payoff in this coalition structure, it is not stable due to secondary clouds 2's deviation.

4.2.4.2 Superadditivity and Externality

In this part, we show the superadditivity and externality of the CFG-e. We consider 4 secondary clouds and 1 primary cloud. Similarly, since we do not intend to evaluate the influence of the parameter values, we assign arbitrary values for the parameters in this part. The primary cloud's capacity is set as 50. The penalty rates of the primary and secondary clouds are set as in the above part. The capacity, request and price vectors of the first three secondary clouds are also set as in the above. The unit coalition cost c^{co} is set as 0 so that cooperation incurs no coalition cost. The capacity of secondary cloud 4 is set as 70. For different number of requests of secondary cloud 4, we evaluate 4 coalition structures: 1) $\{1, 2\}, \{3, 4\}$; 2) $\{1, 2\}, \{3\}, \{4\}$; 3) $\{1\}, \{2\}, \{3, 4\}$; 4) $\{1\}, \{2\}, \{3\}, \{4\}$.

In Fig. 4.4a, we show the values of coalitions $\{3, 4\}$, $\{3\}$ and $\{4\}$ when secondary clouds 1 and 2 cooperate or not. It shows that for all the three coalitions, the coalition value is always larger when secondary clouds 1 and 2 cooperate, which is consistent with the positive externality. Furthermore, for a certain cooperation status of secondary clouds 1 and 2, the value of coalition $\{3, 4\}$ is always larger than the sum of values of coalitions $\{3\}$ and $\{4\}$, which implies superadditivity. The same holds for Fig. 4.4b. In addition, we can see that in Fig. 4.4a, the values of coalitions $\{4\}$ and $\{3, 4\}$ increase

with the number of requests of secondary cloud 4. While this observation is straightforward, we also notice that in both Figs. 4.4a and 4.4b, the values of all the other coalitions which do not contain secondary cloud 4 decrease. This is because when the number of requests in secondary cloud 4 increases, it tends to outsource more requests to the primary cloud, which leads to congestion in the primary cloud and further deteriorates the task performance of other coalitions. This phenomenon indicates that the externality is caused by the resource contention in the primary cloud.

Figure 4.4: Superadditivity & externality

4.2.4.3 Coalition Formation Results under Different Cloud Network Conditions

Figure 4.5: Coalition formation results under different cloud network conditions

This part investigates several key factors that determine the coalition formation results. To approach the parameter values in real scenarios, in this part, we follow the pricing schema of Amazon EC2 [143], and assume that the clouds in the VHCF system use the t2.large VM in a Linux/UNIX operating system, with a per hour price of 0.104 US dollars (i.e., for all primary clouds $i \in \mathcal{I}$, $p_i^p = 0.104$). We consider 2 primary

clouds and 6 secondary clouds. Each secondary cloud j 's capacity is randomized within $[60, 100]$. Its request is randomized within $[c_i^s - 20, c_i^p + 60]$. Its price is randomized within $[p^s - 0.05, p^s + 0.05]$, where p^s indicates the overall price level of the secondary clouds and is randomized within $[0.15, 0.3]$. The price level is 1.5 to 3 times as much as the price of the primary cloud. Considering that we have not counted other costs such as labor, advertisement, and taxes, this is a reasonable range according to the definition of a "healthy operating profit margin" by Investopedia [144]. Each primary cloud i 's capacity is randomized within $[40, 80]$. The penalty rate β^p of the primary clouds is randomized within $[0.08, 0.12]$. The penalty rate of the secondary clouds is set as $\beta^s = \eta p^s$, where η denotes the ratio of penalty against the price level and is randomized within $[0.8, 1.2]$ — by $\eta = 1$, it means that if the processing time is twice as much as the pre-contracted deadline, the penalty equals the price (which means the user does not need to pay for the request). The unit coalition cost c^{co} and the coalition size factor γ are randomized within $[0, 1.2]$ and $[1, 1.3]$, respectively.

In this set of experiments, our strategy is to step by step vary the parameter value under evaluation, and randomize the values of other parameters over the above specified range. For each set of results in Fig. 4.5, we take the average over 30 trials.

We first evaluate the coalition formation results under different coalition cost. We set the unit cost from 0 to 1.5 with a step of 0.3, and observe coalition formation with different coalition size factor γ . Fig. 4.5a shows the average number of coalitions formed under different coalition cost, where smaller number of coalitions means that the secondary clouds tend to cooperate. The figure indicates that when the unit coalition cost and the coalition size factor γ increase, the number of coalitions also increases. When coalition cost approaches 1.5, cooperation is too costly to form. Thus all secondary clouds tend to stay in singleton coalitions.

We then evaluate the coalition formation under different price and penalty rates of the secondary clouds. We set price level p^s of the secondary clouds from 0.15 to 0.3 with a step of 0.03, and evaluate the coalition formation with different secondary clouds' penalty rate η . As is shown in Fig. 4.5b, for a fixed η , the increase of price will promote the cooperation of the secondary clouds. This is because when the price of the secondary clouds increases, the coalition cost is less comparable with the profit.

Figure 4.6: Total utility and total penalty of the secondary clouds

Therefore, cooperation will generate more benefit. Meanwhile, for a fixed price, when penalty rate increases, the secondary clouds tend to cooperate. As is indicated in the proof of Lemma 4.8, cooperation can always lead to no worse delay. Therefore, secondary clouds tend to cooperate to reduce the high delay penalty.

To investigate how the demand and supply relation influences coalition formation, we vary the request scale and capacity of the primary clouds. We define the request scale as the total number of requests received by the secondary clouds minus the total capacity of the secondary clouds. We set it from 40 to 240 with a step of 40, and evaluate its effect on coalition formation with different capacity scale of the primary clouds. Fig. 4.5c shows that when the secondary clouds have smaller capacity or there are more requests from the end users, the secondary clouds tend to cooperate. This is because when the total workload increases or the capacity of the primary clouds decreases, the delay of processing time will be larger. Therefore, the secondary clouds cooperate to share resources and reduce delay.

4.2.4.4 Total Utility Comparison

In this part, we answer the last question in Chapter 1: is it beneficial for the secondary clouds to use a VHCF network? We compare the secondary clouds' total utility and total delay penalty with different request scale for the following 4 cases: 1) VHCF network; 2) no cooperation (the case in [29]); 3) grand coalition (the common practice of horizontal cloud federation); 4) optimal utility (workload factoring and coalition formation are optimized by a mediator). We consider 2 primary clouds and 5 secondary

clouds. Except the capacity and request scales, all the parameter values (please refer to Table 4.3) are set as in the above. We fix the capacity scale of the primary clouds as 100, and investigate different request scales from 40 to 160.

As shown in Figs. 4.6a and 4.6b, the least pessimistic core generates larger total utility than that of no-cooperation and grand coalition. The total utility is higher when the workload scale is closer to the capacity scale of the primary clouds because when the workload is low, the gross revenue is low (although there is a bonus of finishing the requests in advance), but when the workload is too high, the penalty delay is too large. By using the VHCF network, the secondary clouds (using the t2.large VM from Amazon EC2 [143], with an operating scale of 40 to 160 VMs) can averagely decrease the delay penalty by 10.6% and 11.0%, compared with no-cooperation and grand coalition, respectively. The strategic decision making of the secondary clouds naturally leads to a sub-optimal total utility. Unfortunately, the pessimistic core bears an inevitable loss compared with the optimal case, since each cloud provider is self-interested.

4.2.4.5 Compare Different Algorithms

Figure 4.7: Runtime and least core values of different algorithms. A 90% confidence interval is shown.

Last, we compare the runtime and optimality of different algorithms. In this part, all algorithms are implemented using Java, all linear programs are solved with CPLEX (v12.6) and all non-linear programs are solved with KNITRO (v9.0.0). All computations are performed on a 64-bit machine with 16 GB RAM and a quad-core Intel i7-4770 3.4 GHz processor. We consider 2 primary clouds and vary the number of secondary

clouds. All the parameters are randomized as in Section 4.2.4.3. All the results are averaged with 20 trials.

Fig. 4.7a shows the runtime of Brute-Force with CLONE under different convergence threshold e , from 0.001 to 0.00001. Intuitively, smaller e leads to longer runtime, which increases rapidly with the number of secondary clouds. Fig. 4.7b shows the runtime of Brute-Force and CUBE (with different pruning thresholds θ) with CLOSE. We can see that comparing with Fig. 4.7a, the runtime of Brute-Force and CUBE with CLOSE is significantly reduced. When the total number of secondary clouds is smaller than 8, both of the two CLOSE-based algorithms take less than 1 second to finish. Furthermore, by implementing CUBE, the runtime has significantly reduced comparing with Brute-Force, while maintaining a close-to-optimal least ε value (Fig. 4.7c). With smaller θ , it takes less time because more nodes are pruned during the searching. By setting different pruning threshold, we can tradeoff between runtime and optimality. Using these algorithms, we can solve practical scale problems.

4.3 Chapter Summary

The integration of vertical and horizontal cloud federation is a promising approach to improve a cloud's service quality. Notwithstanding the merits, the economic aspects of such a VHCF network is rarely investigated, especially when the secondary clouds are self-interested. With this work, we make the first attempt to reveal the interrelated workload factoring and coalition formation among secondary clouds, and answer the following intrinsic questions: 1) What is the most stable coalition structure and payoff division; 2) What are the optimal (equilibrium) workload factoring strategies of different coalitions; 3) Is it beneficial for the secondary clouds to join such a VHCF network? The experimental results show that interestingly, the two common practices in the real world - no-cooperation and the grand coalition - are sometimes not the most stable co-operation pattern. Instead, the most stable coalition structure usually lies between these two extreme cases. Furthermore, the experimental results show that by implementing the VHCF network, the service delay penalty of the secondary clouds decreases by around 11% compared with the two common practices. The runtime and optimality

evaluation of the algorithms shows that our proposed algorithms can be applied to real world sized problems, and assist secondary clouds to choose their partners and decide the optimal workload factoring strategies. Our framework can be extended to a variety of network-related domains such as cellular networks, cognitive radio networks and smart grid, where cooperative and competitive behaviors coexist.

Chapter 5

Sequential Decision Making with Dynamism and Uncertainties

In the previous two chapters, we focused on strategic decision making in a static environment, where only one-shot decisions are made. In the following two chapters, we look into scenarios where there are dynamism and uncertainties, and a sequence of decisions need to be made. We first study sequential decision making in a single agent environment, which is motivated by the Electronic Toll Collection (ETC) problem and a cooking assistance problem.

Nowadays, governments face a worsening problem of traffic congestion in urban areas. To alleviate road congestion, a number of approaches have been proposed, among which ETC has been reported to be effective in many countries and areas (e.g., Singapore [53], Norway [54]). The tolls of different roads and time periods are different, so that the vehicles are indirectly regulated to travel through less congested roads with lower tolls. However, although current ETC schemes vary tolls at different time periods throughout a day, they are *pre-determined* and *fixed* at the same periods from day to day. A few dynamic road pricing schemes [145–147] have been proposed in the transportation research community which consider the variations of traffic demands over time. However, these tolling schemes still assume that traffic demands are fixed and are known a priori, and thus are *static* in essence. In practice, traffic demands fluctuate and cannot be predicted accurately, especially when the traffic is abnormal (e.g., in case of

traffic accidents, or city events). As a result, these static tolling schemes usually have limited regulation effect. One recent work of Sharon et al. [148] proposes a dynamic tolling scheme called Δ -tolling, which assigns a toll to each road proportional to the difference between its current travel time and its free-flow travel time. However, Δ -tolling does not take a proactive approach towards changes in the demand side. Instead, Δ -tolling only reacts to such changes once they are detected. In contrast, we propose a novel dynamic tolling scheme which optimizes traffic over the long run. Though several existing methods have been proposed to solve the traffic assignment problem with MDP [149, 150], our method is notably distinct from these works, in the sense that they consider the uncertainty of drivers' route choice behavior, while this work considers the uncertainty of traffic demand as well.

With the rapid increase of population with Alzheimer's disease, it has been more and more costly for the society to provide personal care to the patients. Artificial intelligence (AI) technologies, which are believed to significantly reduce the personal care cost, have been widely introduced to assist home care to the elderly people. In this paper, we design a reminder agent to help the patients with Alzheimer's disease in cooking tasks. The reminder agent divides the cooking task into a set of subtasks, and optimizes the time a reminder is released to the patient in each subtask. The cooking procedure is formulated as an MDP with continuous action space. Due to the large scale of the formulated MDP, we propose a solution to the MDP based on reinforcement learning.

The rest of the chapter is organized as follows. Section 5.1 describes the dynamic ETC system model and its formulation as an MDP. Section 5.2 presents the solution algorithm to the formulated MDP. Section 5.3 conducts extensive experiments to evaluate the performance of our proposed DyETC scheme. Section 5.4 introduces another scenario where we propose a solution framework for the automated cooking assistance to patients with the Alzheimer's disease. Section 5.5 summarizes the chapter.

Figure 5.1: Road network of Singapore Central Region

5.1 Formulation of the DyETC Problem

In this section, we first introduce the dynamic ETC system, and then formulate the DyETC problem as an MDP.

5.1.1 Dynamic ETC System

The urban city area can be abstracted as a directed *road network* $G = (E, Z, O)$, where E is the set of roads, Z is the set of zones and O is the set of origin-destination (OD) pairs. Take Singapore Central Region as an example (Figure 5.1), there are altogether 11 zones and 40 abstracted roads.¹ The decision time horizon H (usually the length of rush hour) is discretized into several intervals, with a length of τ (e.g., 10 minutes) for each interval. For time period $t = 0, 1, \dots, H$, we denote an OD pair as a tuple $\langle z_i, z_j, q_{i,j}^t, P_{i,j} \rangle$, where zone z_i is the origin, zone z_j is the destination, $q_{i,j}^t$ denotes traffic demand during time t , and $P_{i,j}$ denotes the set of all possible paths from z_i to z_j which do not contain a cycle. Different from most previous works which assume that the OD traffic demand $q_{i,j}^t$ for a certain time period t is fixed and known *a priori*, we consider dynamic OD travel demand, where the travel demand of an OD pair follows a probability distribution function (PDF) $q_{i,j}^t \sim f(q_{i,j}^t)$.

¹Each pair of two adjacent zones has two directed roads. If there are multiple roads from one zone to another, they could be treated as one abstracted road, where the capacity of the abstracted road is a sum of these roads, and the length of the abstracted road is an average of these roads.

We follow the commonly used *travel time model* [151] to define the travel time on a road e at t :

$$T_e^t = T_e^0 [1 + A(s_e^t/C_e)^B]. \quad (5.1)$$

s_e^t is the number of vehicles on road e . C_e and T_e^0 are road-specific constants, where T_e^0 is interpreted as the free-flow travel time, and C_e is the capacity of the road. A and B are constants which quantify the extent to which congestion influences travel time. Consequently, average travel speed is $\frac{L_e}{T_e^t} = \frac{L_e}{T_e^0 [1 + A(s_e^t/C_e)^B]}$, where L_e is the length of road e . At time t , the travel cost of a path $p \in P_{i,j}$, which we denote as $c_{i,j,p}^t$, consists of both time and monetary costs:

$$c_{i,j,p}^t = \sum_{e \in p} (a_e^t + \omega T_e^t), \quad (5.2)$$

where a_e^t is the toll imposed on road e , and ω is a constant which reveals the value of time. To make the analysis tractable, we assume that all vehicles have the same value of time. Given the current traffic condition (i.e., number of vehicles on each road) and tolls, each vehicle will select a path $p \in P_{i,j}$ leading to its destination, which aggregately forms a *traffic equilibrium*. To describe this traffic equilibrium, we adopt a widely-used *stochastic user equilibrium* (SUE) model [152–154], where the portion of traffic demand $x_{i,j,p}^t$ travelling with path $p \in P_{i,j}$ is

$$x_{i,j,p}^t = \frac{\exp\{-\omega' c_{i,j,p}^t\}}{\sum_{p' \in P_{i,j}} \exp\{-\omega' c_{i,j,p'}^t\}}. \quad (5.3)$$

ω' is a constant measuring vehicles' sensitivity to travel cost.

5.1.2 An MDP Formulation

In general, the government sets the tolls for the current time period and announces them to the vehicles, while each vehicle individually selects paths according to the total travel cost, and the aggregate choice of all the vehicles is assumed to follow the discrete choice model in Eq.(5.3). We formulate the DyETC problem as a discrete time MDP, due to its advantages in modeling sequential planning problems.

Figure 5.2: Event timeline of two subsequent time periods

State & action. At the beginning of time period t , the *state* is defined as the number of vehicles $s_{e,j}^t$ on each road e that are going to destination z_j . $\mathbf{s}_e^t = \langle s_{e,j}^t \rangle$ is the state vector of a road e , and $\mathbf{s}^t = \langle \mathbf{s}_e^t \rangle$ is the state matrix of the road network G . At time t , the government's *action* is to set the tolls $\mathbf{a}^t = \langle a_e^t \rangle, e \in E'$, where $E' \subseteq E$ is the subset of roads which have ETC gantries.

Since both the traffic condition and tolls change over time, a vehicle has an incentive to change its path once it reaches the end of a road. The path readjustment does not depend on the past decisions of the vehicle, but only depends on its destination. Therefore, for a vehicle that arrives at the end z_i of a road e , we treat it as a vehicle from the new origin z_i , while maintaining its destination z_j . To distinguish these vehicles with those that really use z_i as the origin, we define: At time period t , the **primary OD demand** $q_{i,j}^t$ from z_i to z_j is the number of vehicles that originate from z_i at time period t ; while the **secondary OD demand** $\bar{q}_{i,j}^t$ is the number of vehicles that come from z_i 's neighbouring roads during time period $t - 1$ that are heading for destination z_j .

We refer to Figure 5.2 as an illustration of the event timeline for two subsequent time periods. At the beginning of time period t , tolls are decided based on the state $\mathbf{s}^t$ of the current time period. After the tolls are announced to the vehicles, the vehicles will react to the tolls and traffic conditions and a SUE will gradually form during time period t . In practice, it usually takes time to form an SUE and it keeps evolving over time. Before the SUE is formed, the number of vehicles on a road is normally larger than that in the SUE, while after the SUE, the number is usually smaller (and vice versa). To

make the analysis tractable, we use the number of vehicles when the SUE is formed to approximate the average number of vehicles on a road during time period t .

State transition. After the SUE is formed, the state of the next time period can be derived. At the beginning of time $t + 1$, the number of vehicles on road e is determined by the number of vehicles that 1) stay on road e , 2) exit road e , and 3) enter road e during time t . Formally,

$$s_{e,j}^{t+1} = s_{e,j}^t - s_{e,j,out}^t + s_{e,j,in}^t. \quad (5.4)$$

We make a mild and intuitive assumption that the number of vehicles $s_{e,j,out}^t$ that exit a road e is proportional to the average travel speed during time period t . Thus,

$$s_{e,j,out}^t = s_{e,j}^t \cdot \frac{v_e^t \cdot \tau}{L_e} = \frac{s_{e,j}^t \tau}{T_e^0 [1 + A(s_e^t / C_e)^B]}, \quad (5.5)$$

where L_e is the length of road e .

We now derive the last term $s_{e,j,in}^t$ in Eq.(5.4). Recall that the total demand of an OD pair is the sum of primary and secondary OD demand. The secondary demand of an OD pair from z_i to z_j is

$$\bar{q}_{i,j}^t = \sum_{e^+ = z_i} s_{e',j,out}^t, \quad (5.6)$$

where e^+ is the ending point of road e (correspondingly, we use e^- to denote the starting point of e). Note that during time t , for an OD traffic demand from z_i to z_j to be counted as a component of $s_{e,j,in}^t$, two conditions must be satisfied. First, z_i should be the starting point of e , i.e., $z_i = e^-$. Second, at least one of the paths from z_i to z_j should contain road e , i.e., $e \in p \in P_{i,j}$. Thus, we have:

$$s_{e,j,in}^t = \sum_{z_i = e^- \cap e \in p \in P_{i,j}} (q_{i,j}^t + \bar{q}_{i,j}^t) \cdot x_{i,j,p}^t. \quad (5.7)$$

Combining Eqs.(5.4)-(5.7), there is:

$$s_{e,j}^{t+1} = s_{e,j}^t - \frac{s_{e,j}^t \tau}{T_e^0 [1 + A(s_e^t / C_e)^B]} + \sum_{z_i = e^- \cap e \in p \in P_{i,j}} (q_{i,j}^t + \sum_{e' += v_i} s_{e',j,out}^t) \cdot x_{i,j,p}^t. \quad (5.8)$$

Value function & Policy. From the perspective of the government, a good traffic condition means that, during the planning time horizon H , the total traffic volume (i.e., the number of vehicles that reach their destinations) is maximized.¹ Formally, we define the *immediate reward function* $R^t(\mathbf{s}^t, \mathbf{a}^t)$ as the number of vehicles that arrive at their destinations during time t :

$$R^t(\mathbf{s}^t) = \sum_{e \in E} \sum_{z_j = e^+} \frac{s_{e,j}^t \tau}{T_e^0 [1 + A(s_e^t / C_e)^B]}. \quad (5.9)$$

Note that we simplify the notation as $R^t(\mathbf{s}^t)$ since it does not depend on $\mathbf{a}^t$. The long term value function $v^t(\mathbf{s}^t)$ is the sum of rewards from t to $t + H$:

$$v^t(\mathbf{s}^t) = \sum_{t'=t}^{t+H} \gamma^{t'-t} R^{t'}(\mathbf{s}^{t'}), \quad (5.10)$$

where γ is a discount factor. At time t , a *policy* $\pi^t(\mathbf{a}^t | \mathbf{s}^t)$ is a function which specifies the conditional probability of taking an action $\mathbf{a}^t$, given a certain state $\mathbf{s}^t$. The optimal policy maximizes the value function in Eq.(5.10):

$$\pi^{t,*}(\mathbf{a}^t | \mathbf{s}^t) = \arg \max_{\pi^t} v^t(\mathbf{s}^t). \quad (5.11)$$

5.2 Solution Algorithm: PG- β

It is very challenging to find the optimal policy function for the formulated DyETC problem due to three reasons. First, the state space is multi-dimensional w.r.t. the number of roads and destinations. Second, the action space is also multi-dimensional w.r.t. the number of ETC gantries. Moreover, the action space is bounded and continuous.

¹DyETC can be extended to optimize other objective functions, total travel time among them, by changing the reward function appropriately. We leave such extensions to future work.

Last, both the state and action values are dependent on the specific time periods. As a result, it is intractable to find the optimal policy function by simply going through all combinations of state-action pairs, which would be on an astronomical order.

Although numerous reinforcement learning algorithms have been proposed to solve MDPs, they cannot be directly applied to our problem due to the complexities presented above. While the policy gradient methods [19, 94] have shown promise in solving large scale MDPs with continuous action spaces, current policy gradient methods usually focus on MDPs with unbounded action spaces. In the following, we present our solution algorithm, PG- β (Policy Gradient method with Beta distribution based and time-dependent policy functions), with novel improvements to typical applications of policy gradient methods.

5.2.1 General Framework of PG- β

Algorithm 9: PG- β

```

1 Initialize  $\boldsymbol{\vartheta}^t \leftarrow \boldsymbol{\vartheta}_0, \boldsymbol{\theta}^t \leftarrow \boldsymbol{\theta}_0, \forall t = 0, 1, \dots, H$ ;
2 repeat
3   Generate an episode  $\mathbf{s}^0, \mathbf{a}^0, R^0, \dots, \mathbf{s}^H, \mathbf{a}^H, R^H$ ;
4   for  $t = 0, \dots, H$  do
5      $Q^t \leftarrow \sum_{\tau=t}^H R^\tau$ ;
6      $\delta \leftarrow Q^t - \hat{v}(\mathbf{s}^t, \mathbf{a}^t, \boldsymbol{\vartheta}^t)$ 
7      $\boldsymbol{\vartheta}^t \leftarrow \boldsymbol{\vartheta}^t + \beta \delta \nabla_{\boldsymbol{\vartheta}^t} \hat{v}(\mathbf{s}^t, \boldsymbol{\vartheta}^t)$ ;
8      $\boldsymbol{\theta}^t \leftarrow \boldsymbol{\theta}^t + \beta' \delta \nabla_{\boldsymbol{\theta}^t} \log \pi^t(\mathbf{a}^t | \mathbf{s}^t, \boldsymbol{\theta}^t)$ 
9 until #episodes =  $M$ 
10 return  $\boldsymbol{\theta}^t, \forall t = 0, 1, \dots, H$ ;

```

The idea of PG- β is to approximate the policy function with a parameterized function, and update the parameter with stochastic gradient descent method. More specifically, PG- β incorporates an *actor-critic* architecture, where the “actor” is the learned policy function which performs action selection, and the “critic” refers to the learned value function measuring the performance of the current policy function.

In Algorithm 9, the input of PG- β includes the planning horizon H , the state transition function Eq.(5.8), a set of parameterized value functions $v^t(\mathbf{s}, \boldsymbol{\vartheta}^t)$ w.r.t. parameter

$\boldsymbol{\vartheta}^t$ ($\forall t = 0, \dots, H$), and a set of parameterized policy functions $\pi^t(\mathbf{a}|\mathbf{s}, \boldsymbol{\theta}^t)$ w.r.t. parameter $\boldsymbol{\theta}^t$ ($\forall t = 0, \dots, H$). The algorithm starts with an initialization of the parameters $\boldsymbol{\vartheta}^t$ and $\boldsymbol{\theta}^t$. It then enters the repeat loop (Lines 2-8). In the repeat loop, it first simulates an *episode*, which contains a series of state-action pairs in the planning horizon H . The states are generated following the state transition function Eq.(5.8), the actions are selected following the policy function $\pi(\mathbf{a}|\mathbf{s}, \boldsymbol{\theta})$, while the immediate rewards are computed with Eq.(5.9). After an episode is simulated, the algorithm then updates the parameters at each time period $t = 0, \dots, H$ (Lines 4-8) with stochastic gradient descent method. Q^t denotes the sum of rewards from time t to H obtained from the simulated episode, which reveals the “real” value obtained by the current policy, while $\hat{v}^t(\mathbf{s}^t, \boldsymbol{\vartheta}^t)$ is the “estimated” sum of rewards approximated by the parameterized value function $v^t(\mathbf{s}^t, \boldsymbol{\vartheta}^t)$. Consequently, δ denotes the difference of these two terms. In Lines 6-7, PG- β updates the parameters by our derived update rule which is to be introduced in the two subsequent subsections. This process iterates until the number of episodes reaches a pre-defined large number M (e.g., 100,000).

5.2.2 Policy Gradient for Time-Dependent MDP

A fundamental property of traditional MDPs is that the value of a state does not change over time. However, such property does not exist in our problem. Intuitively, with the same number of vehicles on the road network, the number of vehicles that reach the destinations (i.e., the objective) still depends on the OD demand of a specific time period as well as future periods. Consequently, the value of an action is also dependent on the specific time period. This class of MDPs is called *finite horizon MDPs* (FH-MDPs). For FHMDPs, we need to maintain and update a value function $v^t(\mathbf{s}, \boldsymbol{\vartheta}^t)$ and a policy function $\pi^t(\mathbf{a}|\mathbf{s}, \boldsymbol{\theta}^t)$ for each time step $t = 0, 1, \dots, H$, as shown in Algorithm 9. The following theorem ensures that the update of $\boldsymbol{\theta}^t$ improves action selection in the FHMDP.¹

¹For FHMDPs, a natural alternative is to incorporate the time as one extra dimension of state. However, experimental results (see Figure 5.3a) show that this alternative does not work well using polynomial basis functions where states are not interrelated. Other alternatives might be to utilize an intrinsic basis function, or use deep neural networks to represent the policy functions. However, such intrinsic functions are very hard to find (perhaps it does not exist), while the time consumed in training an effective neural network is usually way longer than training linear policy functions.

Theorem 5.1. *The gradient function of policy gradient method on FHMDPs is*

$$\nabla_{\theta^t} v_{\pi}(\mathbf{s}) = Q^t(\mathbf{s}^t, \mathbf{a}^t) \nabla_{\theta^t} \log \pi^t(\mathbf{a}^t | \mathbf{s}^t, \theta^t), \quad (5.12)$$

where $Q^t(\mathbf{s}^t, \mathbf{a}^t)$ is the action value of $\mathbf{a}^t$ given state $\mathbf{s}^t$.

Proof. We first write the state value as the expected sum of action values, i.e., $v^t(\mathbf{s}^t) = \sum_{\mathbf{a}^t} \pi^t(\mathbf{a}^t | \mathbf{s}^t, \theta^t) Q^t(\mathbf{s}^t, \mathbf{a}^t)$, where $Q^t(\mathbf{s}^t, \mathbf{a}^t) = \sum_{\mathbf{s}^{t+1}} P(\mathbf{s}^{t+1}, \mathbf{a}^t, \mathbf{s}^t) v^{t+1}(\mathbf{s}^{t+1})$. Since both the transition probability function $P(\mathbf{s}^{t+1}, \mathbf{a}^t, \mathbf{s}^t)$ and the value function $v^{t+1}(\mathbf{s}^{t+1})$ of time $t + 1$ do not depend on θ^t , $Q^t(\mathbf{s}^t, \mathbf{a}^t)$ is also independent on θ^t . Thus, we have

$$\begin{aligned} \nabla_{\theta^t} v^t(\mathbf{s}^t) &= \sum_{\mathbf{a}^t} Q^t(\mathbf{s}^t, \mathbf{a}^t) \nabla_{\theta^t} \pi^t(\mathbf{a}^t | \mathbf{s}^t, \theta^t) \\ &= \sum_{\mathbf{a}^t} \pi^t(\mathbf{a}^t | \mathbf{s}^t, \theta^t) Q^t(\mathbf{s}^t, \mathbf{a}^t) \frac{\nabla_{\theta^t} \pi^t(\mathbf{a}^t | \mathbf{s}^t, \theta^t)}{\pi^t(\mathbf{s}^t, \mathbf{a}^t)} \\ &\quad (\text{multiplying and dividing by } \pi^t(\mathbf{a}^t | \mathbf{s}^t, \theta^t)) \\ &= E[Q^t_{\pi}(\mathbf{s}^t, \mathbf{a}^t) \frac{\nabla_{\theta^t} \pi^t(\mathbf{a}^t | \mathbf{s}^t, \theta^t)}{\pi^t(\mathbf{a}^t | \mathbf{s}^t, \theta^t)}]. \end{aligned}$$

For stochastic gradient, a sampled action $\mathbf{A}^t$ is used to replace the expectation, i.e.,

$$\begin{aligned} \nabla_{\theta^t} v^t(\mathbf{s}^t) &= Q^t_{\pi}(\mathbf{s}^t, \mathbf{A}^t) \frac{\nabla_{\theta^t} \pi^t(\mathbf{A}^t | \mathbf{s}^t, \theta^t)}{\pi^t(\mathbf{A}^t | \mathbf{s}^t, \theta^t)} \\ &= Q^t(\mathbf{s}^t, \mathbf{A}^t) \nabla_{\theta^t} \log \pi^t(\mathbf{A}^t | \mathbf{s}^t, \theta^t) \end{aligned}$$

□

5.2.3 A Novel Policy Function Based on the Beta PDF

An important challenge of the policy gradient method is to balance the “exploitation” of the optimal action generated from the policy function and the “exploration” of the action space to ensure that the policy function is not biased. For MDPs with *continuous* action space, a Normal PDF has been used in recent works [155] as the policy function. While Normal PDF works well in cases where the action space is unbounded, it is not suitable for MDPs which have bounded action spaces, since the action generated by the Normal PDF policy function would possibly become infeasible. In practice, tolls

are restricted within a certain interval $[0, a_{max}]$ (e.g., in Singapore, tolls are within 6 Singapore dollars). A straightforward adaptation is to project the generated action to the feasible action space whenever it is infeasible. However, experimental results (will be presented later in Figure 5.3a) show that, even with such adaptation, Normal PDF policy function performs poorly in solving MDPs with bounded action spaces.

To adapt policy gradient methods to bounded action space, we propose a new form of policy function, which is derived from Beta PDF $f(x)$:

$$f(x, \lambda, \nu) = \frac{x^{\lambda-1}(1-x)^{\xi-1}}{B(\lambda, \xi)}, \quad (5.13)$$

where the *Beta function*, $B(\lambda, \xi) = \int_0^1 t^{\lambda-1}(1-t)^{\xi-1}dt$ is a normalization constant, and the variable $x \in [0, 1]$ is bounded and continuous. For each road e , let $x_e = a_e/a_{max}$, then the policy function is denoted as

$$\pi_e(a_e = x_e a_{max} | \mathbf{s}, \lambda_e, \xi_e) = \frac{x_e^{\lambda_e-1}(1-x_e)^{\xi_e-1}}{B(\lambda_e, \xi_e)}, \quad (5.14)$$

where $\lambda_e = \lambda_e(\phi(\mathbf{s}), \boldsymbol{\theta}^\lambda)$ and $\xi_e = \xi_e(\phi(\mathbf{s}), \boldsymbol{\theta}^\xi)$ are parameterized functions, and $\boldsymbol{\theta}^\lambda$ and $\boldsymbol{\theta}^\xi$ are parameters to be approximated. For example, in the commonly utilized *linear function approximation*, each λ_e and ξ_e is represented by a linear function of the parameters $\boldsymbol{\theta}^\lambda$ and $\boldsymbol{\theta}^\xi$: $\lambda_e = \boldsymbol{\theta}^\lambda \cdot \phi(\mathbf{s})$, $\xi_e = \boldsymbol{\theta}^\xi \cdot \phi(\mathbf{s})$.

Proposition 5.2. *Using Beta PDF policy function, the update rule for a parameter $\theta_{e,e',j,i}^\lambda$ (associated with road e) is*

$$\begin{aligned} \theta_{e,e',j,i}^\lambda &\leftarrow \theta_{e,e',j,i}^\lambda + \beta' \delta [\ln(x_e) - \Psi(\lambda_e) + \Psi(\lambda_e + \xi_e)] \frac{\partial \lambda_e}{\partial \theta_{e,e',j,i}^\lambda}, \\ \theta_{e,e',j,i}^\xi &\leftarrow \theta_{e,e',j,i}^\xi + \beta'' \delta [\ln(1-x_e) - \Psi(\xi_e) + \Psi(\lambda_e + \xi_e)] \frac{\partial \xi_e}{\partial \theta_{e,e',j,i}^\xi}, \end{aligned}$$

where β' and β'' are learning rates for θ^λ and θ^ξ , respectively, $\Psi(\cdot)$ is the digamma function. e' , j and i respectively correspond to a road, a destination and a dimension in the basis functions.

Proof. For ease of notation, we discard the superscript t . In Eq.(5.12), for edge e and dimension i of θ^λ , by substituting $\pi(\mathbf{a}|\mathbf{s}, \theta)$ with Eq.(5.14), we have

$$\begin{aligned}
\frac{\partial v(\mathbf{s})}{\partial \theta_{e,e',j,i}^\lambda} &= \frac{Q(\mathbf{s}, \mathbf{a}) \partial \pi_e(a_e|\mathbf{s}, \theta_{e,e',j,i}^\lambda)}{\pi_e(a_e|\mathbf{s}, \theta_{e,e',j,i}^\lambda) \partial \theta_{e,e',j,i}^\lambda} \\
&= \frac{Q(\mathbf{s}, \mathbf{a})}{\pi_e(a_e|\mathbf{s}, \theta_{e,e',j,i}^\lambda)} \frac{\partial \pi_e(a_e|\mathbf{s}, \theta_{e,e',j,i}^\lambda)}{\partial \lambda_e} \frac{\partial \lambda_e}{\partial \theta_{e,e',j,i}^\lambda} \\
&= \frac{Q(\mathbf{s}, \mathbf{a})}{\pi_e(a_e|\mathbf{s}, \theta_{e,e',j,i}^\lambda)} \frac{\partial \xi_e}{\partial \theta_{e,e',j,i}^\xi} \left[\frac{x_e^{\lambda_e-1} \ln x_e (1-x_e)^{\xi_e-1}}{B(\lambda_e, \xi_e)} - \right. \\
&\quad \left. \frac{x_e^{\lambda_e-1} (1-x_e)^{\xi_e-1} \partial B(\lambda_e, \xi_e) / \partial \lambda_e}{B^2(\lambda_e, \xi_e)} \right] \\
&= \frac{Q(\mathbf{s}, \mathbf{a})}{\pi_e(a_e|\mathbf{s}, \theta_{e,e',j,i}^\lambda)} \frac{\partial \xi_e}{\partial \theta_{e,e',j,i}^\xi} \\
&\quad \frac{x_e^{\lambda_e-1} (1-x_e)^{\xi_e-1} [\ln x_e - \Psi(\lambda_e) + \Psi(\lambda_e + \xi_e)]}{B(\lambda_e, \xi_e)} \\
&= Q(\mathbf{s}, \mathbf{a}) [\ln x_e - \Psi(\lambda_e) + \Psi(\lambda_e + \xi_e)] \frac{\partial \xi_e}{\partial \theta_{e,e',j,i}^\xi}
\end{aligned}$$

By replacing $Q(\mathbf{s}, \mathbf{a})$ with δ , we obtain the update rule for $\theta_{e,e',j,i}^\lambda$. Similarly, we derive the update rule for $\theta_{e,e',j,i}^\xi$. $\square$

5.2.4 State Abstraction

As discussed above, there are two corresponding parameters $\theta_{e,e',j,i}^\lambda$ and $\theta_{e,e',j,i}^\xi$ for each edge e , each edge e' , each destination j and each dimension i in the basis functions. As a result, the total number of parameters in the policy function is of order $\propto |E|^2 |Z| H d$, where $|E|$, $|Z|$, H and d are respectively the number of edges, number of vertices (destinations), length of time horizon and number of dimensions in the basis functions. In this case, when a road network grows too large, it becomes rather time consuming for PG- β to learn an effective tolling policy. Moreover, higher dimension states may lead to over-fitting. To handle these two issues, we conjecture:

Conjecture 1. The vehicles on a same edge that are going to different destinations have almost equal effects on tolls.

This conjecture means that, for the tolls on road e , the parameters (weights) associated to $s_{e',j}$ and $s_{e',i}$ are equal:

$$\theta_{e,e',j,i}^\lambda = \theta_{e,e',j',i}^\lambda = \theta_{e,e',i}^\lambda, \forall j, j' \quad (5.15)$$

$$\theta_{e,e',j,i}^\xi = \theta_{e,e',j',i}^\xi = \theta_{e,e',i}^\xi, \forall j, j' \quad (5.16)$$

The supporting evidence of this conjecture will be shown through experimental evaluations.

5.3 Experimental Evaluation

In this section, we conduct experiments on both simulated settings and a real-world road network in Singapore Central Region to evaluate our proposed DyETC scheme and its solution algorithm PG- β . All algorithms are implemented using Java, all computations are performed on a 64-bit machine with 16 GB RAM and a quad-core Intel i7-4770 3.4 GHz processor.

5.3.1 Evaluation on Synthetic Data

We first conduct experiments on synthetic data. For policy gradient methods, we first obtain the policy function with offline training, and then use the trained policy to evaluate their performance. Unless otherwise specified, the parameters of this subsection are set as follows.

Learning-related parameters. The learning rates of the value function and policy function are hand-tuned as 10^{-7} and 10^{-10} , respectively. The discount factor γ is set as 1, which assigns same weights to rewards of different time periods in the finite time horizon H . The number of episodes for training is 50,000 and the number of episodes for validation is 10,000.

Road network-related parameters. The number of zones in the simulation is set as $|Z| = 5$, and all zones can be destinations, i.e., $|Z'| = |Z|$. The number of roads

Figure 5.3: Performance of different policy gradient methods

is set to 14, and all roads have ETC gantries, i.e., $|E'| = |E| = 14$. The lengths of roads are randomized within $[4, 10]$ km, which is the usual length range between zones of a city. Capacity of a road is set as 50 vehicles per kilometer per lane. This amount is obtained from an empirical study of Singapore roads [156]. Free flow travel speed $\frac{T_e^0}{L_e} = 0.5$ km/min. The parameters in the travel time model Eq.(5.1) is set as $A = 0.15$, $B = 4$ according to [151].

Demand-related parameters. OD demand is simulated as a step function of time, where demand at time $t = 0$ is the lowest, and gradually grows to peak demand in the middle of the planning time horizon, and decreases again to a lower level. The peak demand for each OD pair is randomized within $[8, 12]$ vehicles per minute, which is a reasonable amount. The OD demand at $t = 0$ (which is usually the beginning of rush hour) is set as 60% of the peak demand. Initial state is randomized within $[0.5, 0.7]$ of the capacity of a road.

Toll-related parameters. Maximum toll $a_{max} = 6$. This value is obtained from the current toll scheme in Singapore. Planning horizon $H = 6$. Length of a time period $\tau = 10$ mins. Passenger cost-sensitivity level $\omega' = 0.5$, and value of time $\omega = 0.5$. We will evaluate other values for these two terms.

Comparison of different policy gradient methods.

We compare the learning curve and solution quality of PG-β with the following policy gradient algorithms.

Figure 5.4: Traffic volume (in thousands) of existing tolling schemes under various traffic conditions

Figure 5.5: Total travel time (in thousands) of existing tolling schemes under various traffic conditions

1. PG-N: policy gradient (PG) with Normal distribution based policy function.
2. PG-I: PG with time-independent policy function.
3. PG-time: PG- β where time is incorporated into state.
4. PG- β -abs: PG- β with state abstraction.

Figure 5.3a shows the learning curve of different policy gradient methods, where x-axis is the number of training episodes, y-axis is the traffic volume (in thousands). It shows that PG- β and PG- β -abs converge faster (in terms of number of training episodes) than other policy gradient methods, and achieve higher traffic volume after 50,000 episodes. It is worth mentioning that the classical policy gradient method PG-N cannot learn an effective policy in our problem. Moreover, the learning curves of PG- β and PG- β -abs almost overlap, which gives supporting evidence to Conjecture 1. Figure 5.3b presents per episode runtime (in milliseconds) of different policy gradient methods, where PG- β -abs has the shortest per episode runtime. Combining the two figures, we conclude that PG- β -abs is the best approach in terms of both runtime efficiency and optimality. In the following, we implement PG- β -abs to compare with other existing tolling schemes, while using PG- β to denote PG- β -abs for neatness of notation.

Comparison of PG- β with existing tolling schemes under different settings. We now compare PG- β with the following baseline tolling schemes:

1. Fix: fixed toll proportional to average OD demand.
2. DyState: dynamic toll proportional to the state scale.
3. Δ -toll [148].
4. P0: no tolls.

To compare tolling schemes under different settings, we vary the traffic parameter which under evaluation, and keep all the other parameters as stated above. Figure 5.4 shows the traffic volume obtained by different tolling schemes under different settings, where the y-axis is the traffic volume, and the x-axis is the value of the parameter that is under evaluation. Figures 5.4a-5.4b show that, the traffic volume increases linearly w.r.t. the increasing initial state and OD demand ratio, and PG- β works well under different initial state and OD demand scales. Similarly in Figures 5.4c-5.4d, with a higher cost-sensitivity level and value of time, the traffic volume of all tolling schemes increases. According to Eq.(5.3), this is intuitive in the sense that when the travelling cost of a congested road grows (faster than a less congested road), vehicles diverge towards less congested roads with less travelling cost. When these two parameters grow too large, the regulation effect of all tolling schemes becomes saturated and traffic volume converges to a maximum amount. In this case, PG- β has a larger maximum limit than other tolling schemes. Figure 5.4e shows that PG- β works well with different maximum toll amounts, and it works even better when the maximum toll amount grows. In general, PG- β outperforms existing tolling schemes under all settings. It is worth mentioning that, the state-of-the-art Δ -tolling approach, which does not take a pro-active approach towards changes in the demand side, does not work well in our setting.

To demonstrate that our DyETC framework can be adapted to other objectives, we also evaluate the total travel time of different tolling schemes under the above settings (Figure 5.5, where the y-axis is the total travel time). We can see that PG- β still significantly outperforms all the other tolling schemes (the less total travel time, the better).

5.3.2 Evaluation on a Real-World Road Network of Singapore Central Region

In this subsection, we evaluate the performance of PG- β for its regulation effect on the morning rush hour traffic of Singapore Central Region. The number of episodes for training PG- β is 500,000, and the learning rates for the value and policy functions are fine-tuned as 10^{-8} and 10^{-12} , respectively. Figure 5.6 shows the abstracted road network of Singapore Central Region, where the zones are labelled from 1 to 11. The numbers along the roads denote the travel distance of the adjacent zones, which is obtained from Google Map. Since the OD demand is not revealed by Singapore government, we use the population of different zones to estimate it. The data is obtained from the Department of Statistics [157] of Singapore in 2016. We first obtain the total number of vehicles as 957,246 and the population of Singapore as 5,535,002. The per person vehicle ownership is $956,430/5,696,506 = 0.173$. We then obtain the population of each zone in the Central Region and thus are able to estimate the number of vehicles in each zone (the origin side during morning rush hour). For the destination side, we categorize the 11 zones into 3 types: type 1 is the Downtown Core which is the center of the Central Region, type 2 is the zones that are adjacent to Downtown Core, type 3 is the other zones. We assume a 1 : 0.8 : 0.6 of demand ratio for these three types of zones. We assume 40% of the vehicles in each zone will go to the Central Region. All the other parameters are estimated as those in the above subsection.

As shown in Figure 5.7, when applied to Singapore Central Region, DyETC significantly outperforms the other tolling schemes, in terms of both total traffic volume and total travel time. Compared with the second best tolling scheme (Fix), DyETC is able to increase the traffic volume by around 8% (compared with Fix), and decrease the total travel time by around 14.6% (compared with Δ -tolling).

Figure 5.6: Singapore Central Region road network

Figure 5.7: Performance of existing tolling schemes evaluated in Singapore Central Region

5.4 A Cooking Assistance System for Patients with Alzheimer's disease

This In 2016, it is estimated that nearly 44 million people have Alzheimer's or a related dementia worldwide, and the global cost of caring Alzheimer's and dementia is estimated to be 605 billion, which is equivalent to 1% of the entire world's gross domestic product [158]. This situation is growing worse, as shown in Figure 5.8, the number of people affected by Alzheimer's is expected to grow to around 100 million in the year 2050.

Due to the cost efficiency and high quality care, AI technologies have been gradually adopted to assist daily home care to the patients with Alzheimer's. Cooking has been a critical, and yet challenging task for the patients with Alzheimer's (in the following, we will simply refer to them as "patients"). Take Spaghetti as an example (Figure 5.9), there are several steps to successfully cook Spaghetti, which is complex to the patients without any assistance. In this paper, we propose a cooking assistance agent, which gives guidance to the patients while cooking. More specifically, our agent breaks down a cooking task (e.g., cooking Spaghetti) into several subtasks, and provide a reminder of what to do to the patient at a proper time during the subtask.

We make two key contributions in this project. First, we construct a formal model to formulate the cooking assistance task as an MDP. The state of the MDP is the time (or estimated time) the patient spends on each task, which reveals the mentality status of the patient. The action is to decide, at the beginning of each subtask, when to release a reminder to the patient. To our knowledge, there are no previous papers formulating the cooking assistance procedure as an MDP. Our second key contribution is a solution to the formulated MDP based on inverse reinforcement learning. We show that the formulated MDP has continuous (and thus infinitely large) state and action spaces, which makes it very challenging to solve the MDP. In this paper, we propose a solution which 1) employs function approximation approach to represent the value function, and 2) a policy gradient method which directly finds an optimal policy with a function form.

The first thread of work that we follow is the MDP or POMDP based service assistance agents. In [159], the authors propose a hand-washing assistance agent based on an MDP formulation (and is later generalized into a POMDP [160]). Different from the complex MDP formulated, the MDP formulated in this work has finite number of states and actions and thus cannot be applied to our problem. A more general framework of service assistance agent is proposed in [161]. In this work, the assisted tasks are not specified, and thus cannot be directly applied to solve the domain specific problem of cooking assistance.

We follow the second thread of work of solving large scale MDPs. There have been a great amount of works on solving large scale MDPs. One category of methods

Figure 5.8: Statistics of Alzheimer's Disease

Figure 5.9: Steps of Cooking Spaghetti

focuses on tabular representations of the value and policy functions. These tabular-based methods include tabular tabular Q-learning [15], prioritized sweeping [16], Monte Carlo Tree Search (MCTS) [17] and UCT (UCB applied to trees) [18]). However, the state space of our formulated MDP is continuous, and thus cannot be represented with a tabular form.

Another category of methods use approximated functions to represent the value functions. Traditional function approximation methods [90–92] perform action selection by approximating state-action values. Due to the continuous action spaces, the action selection process becomes rather inefficient with these traditional value function approximation methods. As a result, we adopt a policy gradient method [19, 94] which directly represents the policy as a parameterized function of states.

5.4.1 Formulation

5.4.1.1 Cooking System Formulation

Formally, we consider a cooking task T which consists of a set of sequential subtasks $\{T_1, \dots, T_n\}$. We assume that a subtask $T_i (i \geq 1)$ can be triggered only when its

preceding task T_{i-1} is completed. This is intuitive in a cooking task. Take Spaghetti as an example, the subtask of cooking Spaghetti can be triggered only when the subtask of boiling water is completed.

A cooking assistance agent is able to remind the patient when the patient is performing each subtask. The reminder can be triggered at time t_i during each task, where finish of the last subtask T_{i-1} is set by default as $t_i = 0$. Generally, by triggering a reminder earlier, the patient would have a higher utility by completing the subtask with a shorter time, but he/she will also experience a higher level of frustration. We denote the payoff of the patient completing a subtask $T_i \in T$ as $v_i(t_i)$, which is assumed to be a decreasing function w.r.t. time t_i . We denote the frustration loss of the patient being reminded at time t_i while completing a subtask T_i as $c_i(t_i)$, which is assumed to be a decreasing function of time.

5.4.1.2 MDP of Cooking Assistance Agent

We formulate the planning problem of the assistance agent as an MDP. Formally, the *state* of the MDP is a vector $\mathbf{s} = \langle s_1, \dots, s_n, s_{n+1} \rangle$ where each $s_i = \langle s_1, \dots, s_n \rangle$ represents the time it takes for the patient to accomplish each task T_i , and $s_{n+1} = i$ indicates the subtask T_i the patient is starting to perform. If a subtask T_i has already been completed (i.e., $s_{n+1} > i$), then s_i means the actual task completion time. Otherwise, s_i is the estimated task completion time. Intuitively, the system state reveals the mentality status of the patient, i.e., if the patient is able to complete the subtasks within a shorter time, it is believed that the patient is in a better status, and tends to be able to complete the remaining subtasks faster. We denote the estimated task completion time for each subtask T_i as a probability distribution function of state: $f(t_i, \mathbf{s})$, where t_i^{max} is the longest time a patient can complete subtask T_i . It is required that

$$\int_{t_i=0}^{t_i^{max}} f(t_i, \mathbf{s}) = 1. \quad (5.17)$$

The *action* of the assistance agent is to decide, for each subtask T_i , the time a_i a reminder is released to the patient during the subtask, given the current state $\mathbf{s}$. After

the reminder for subtask T_i is released, we assume that the patient is able to complete the subtask within a fixed time t_i^0 . Note that the actions taken after a subtask T_i is completed will not affect the task completion time of the previous subtasks anymore.

As a result, taken action a_i , the expected utility of completing a subtask T_i (i.e., the *immediate reward*) is denoted as

$$r_i(a_i) = \int_{t_i=0}^{a_i} v_i(t_i) f(t_i, \mathbf{s}) dt_i + \int_{t_i=a_i}^{t_i^{max}} v_i(a_i + t_i^0) f(t_i, \mathbf{s}) dt_i. \quad (5.18)$$

The first integration on the right hand side denotes the expected utility of the patient if he/she completes the subtask T_i before the reminder is released, while the second integration denotes the other case.

The long term reward or *value function* is represented as a sum of all the immediate reward of the tasks completed after subtask T_i :

$$v(\mathbf{s}, a_i) = \sum_{j=i}^n r_j(a_j). \quad (5.19)$$

A *policy* is $\pi(a_i|\mathbf{s})$ is a function which specifies the conditional probability of taking an action a_i , given a certain state $\mathbf{s}$. The optimal policy maximizes the value function in Eq. (5.19):

$$\pi^*(a_i|\mathbf{s}) = \arg \max_{\pi} v(\mathbf{s}, a_i). \quad (5.20)$$

The *state transition function* is the probability of a state $\mathbf{s}$ transiting into state $\mathbf{s}'$, given certain action a_i . In the next state, since only s_i is changed, it is equivalent to the probability of the patient using s_i time to complete the subtask T_i :

$$p(\mathbf{s}'|\mathbf{s}, a_i) = p(s_i|\mathbf{s}, a_i). \quad (5.21)$$

5.4.1.3 State Transition Function

As we can see, the state transition function depends on the probability distribution function $f(t_i, \mathbf{s})$. While our formulation above is general to all types of $f(t_i, \mathbf{s})$, in this

subsection, we study a specific form, i.e., a normal distribution. More specifically, we consider a probability distribution function of the completion time t_i of subtask T_i , given state $\mathbf{s}$:

$$f(t_i, \mathbf{s}) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left\{-\frac{(t_i - \mu_i)^2}{\sigma_i^2}\right\}, \quad (5.22)$$

where $\mu_i = \mu_i(\mathbf{s})$ and σ_i are respectively the mean and standard deviation of the distribution. We assume

$$\mu_i = \bar{\mu}_i \frac{\sum_{j=1}^{i-1} s_j}{\sum_{j=1}^{i-1} \bar{\mu}_j}, \quad (5.23)$$

where $\bar{\mu}_i$ is the average task completion time of subtask T_i obtained from historical statistics. We also assume σ_i can be obtained from historical statistics and does not change over time. Intuitively, this indicates that if the task completion time of the previous subtasks is long, then the task completion time of subtask T_i would also be long.

As a result, we have

$$f(s_i | \mathbf{s}, a_i) = \begin{cases} f(t_i, \mathbf{s}), & s_i \leq a_i; \\ \int_{a_i}^{\infty} f(t_i, \mathbf{s}) dt_i, & s_i = a_i + t_i^0. \end{cases} \quad (5.24)$$

5.4.2 Analysis and Proposed Methods

As we can see, the formulated MDP is a finite horizon MDP with continuous action and state spaces. Solving this formulated MDP is very challenging due to two major reasons. First, the formulated MDP has very large scale action and state spaces. For such large scale MDP, it is difficult to even represent the value functions with a tabular form. As a result, we use an approximated function form to represent the value function. In general, *function approximation* selects a function among a well-defined class that closely matches (“approximate”) a target function in a task-specific way. The novelty of function approximation is that the approximate value function $v(\mathbf{s}, \theta)$ is represented not as a table but as a parameterized functional form with weight vector θ . $v(\mathbf{s}, \theta)$ might be a linear function of the features of the state, with θ as the weight vector of the features. More generally, $v(\mathbf{s}, \theta)$ might be the function computed by a multi-layer artificial neural network, with θ the vector of connection weights in all the layers. By

adjusting the weights, any of a wide range of different functions can be implemented by the network.

Even with function approximation, the search spaces of action and state are still infinitely large, which makes it infeasible for traditional search methods such as Monte Carlo Tree Search (MCTS) [17], UCT (UCB applied to trees) [18] or function approximation methods [90–92] that perform action selection by approximating state-action values cannot be applied to our problem. As a result, we propose a policy gradient method in reinforcement learning to solve the problem.

The idea of policy gradient method is to approximate the policy function with a parameterized function $\pi(\mathbf{a}|\mathbf{s}, \vartheta)$, and update the parameter with the stochastic gradient descent method. Similar to the value function, the form of the approximated policy function can either be linear to the features of the state or an artificial neural network. More specifically, PG- β incorporates an *actor-critic* architecture, where “actor” is the learned policy function which performs action selection, and “critic” refers to the learned value function which is a performance measure of the current policy function. As shown in Algorithm 10, the input of the policy gradient method includes the state transition function, a set of parameterized value functions $v_i(\mathbf{s}, \theta_i), \forall i = 1, \dots, n$ with respect to parameter $\theta_i, \forall i = 1, \dots, n$ and a set of parameterized policy functions $\pi_i(a_i|\mathbf{s}, \vartheta_i), \forall i = 1, \dots, n$ with respect to parameter $\vartheta_i, \forall i = 1, \dots, n$.

More specifically, we adopt the most commonly used form of value function, where the value function is linear with respect to the feature functions $\phi(\mathbf{s})$, i.e.,

$$v_i(\mathbf{s}, \theta_i) = \sum_j \theta_{ij} \phi_j.$$

For the choice of a proper policy function, a key aspect is to trade-off the balance between exploitation (of a good policy) and exploration (of potential better policy). To do this, we use a normal distribution function to represent the policy function, instead of using a deterministic policy function. Formally, we denote the policy function as

$$\pi_i(a_i|\mathbf{s}, \vartheta_i^\sigma, \vartheta_i^\mu) = \frac{1}{\sqrt{2\pi\sigma_i'^2}} \exp\left\{-\frac{(a_i - \mu_i')^2}{\sigma_i'^2}\right\},$$

where σ' and μ' are respectively the standard deviation and mean value of the normal distribution used to sample the action a_i , which are linear functions with respect to the feature functions $\phi(\mathbf{s})$ with the parameters ϑ^σ and ϑ^μ .

Algorithm 10: Policy Gradient

```

1 Initialize  $\vartheta \leftarrow \vartheta_0, \theta \leftarrow \theta_0$ ;
2 repeat
3   Generate an episode  $\mathbf{s}^1, \mathbf{a}^1, R^1, \dots, \mathbf{s}^n, \mathbf{a}^n, R^H$ ;
4    $Q \leftarrow \sum_{i=1}^n r_i$ ;
5    $\delta \leftarrow Q - \hat{v}(\mathbf{s}, \mathbf{a}, \theta)$ 
6    $\vartheta \leftarrow \vartheta + \beta \delta \nabla_{\vartheta} \hat{v}(\mathbf{s}, \vartheta)$ ;
7    $\theta \leftarrow \theta + \beta' \delta \nabla_{\theta} \log \pi(\mathbf{a}|\mathbf{s}, \theta)$ 
8 until forever
9 return  $\theta$ ;

```

The algorithm starts with an initialization of the parameters ϑ and θ . It then enters the repeat loop (Lines 2-8). In the repeat loop, it first simulates an *episode*, which contains a series of state-action pairs in the planning horizon H . The states are generated following the state transition function Eq.(5.21), the actions are selected following the policy function $\pi(\mathbf{a}|\mathbf{s}, \theta)$, while the immediate rewards are computed with Eq.(5.18). After an episode is simulated, the algorithm then updates the parameters (Lines 4-7) with stochastic gradient descent method. Q denotes the sum of rewards of completing all the subtasks obtained from the simulated episode, which reveals the “real” value obtained by the current policy, while $\hat{v}^t(\mathbf{s}^t, \vartheta^t)$ is the “estimated” sum of rewards approximated by the parameterized value function $v^t(\mathbf{s}^t, \vartheta^t)$. Consequently, δ denotes the difference of these two terms. In Lines 6-7, policy gradient method updates the parameters with the specified gradient methods.

5.5 Chapter Summary

In this chapter, we propose the DyETC scheme for optimal and dynamic road tolling in urban road network. We make three key contributions. First, we propose a formal model of the DyETC problem, which is formulated as a discrete-time MDP. Second, we develop a novel solution algorithm, PG- β to solve the formulated large scale MDP.

Third, we conduct extensive experimental evaluations to compare our proposed method with existing tolling schemes. The results show that on a real world traffic network in Singapore, PG- β increases the traffic volume by around 8%, and reduces the travel time by around 14.6% during rush hour. As another application of the proposed policy gradient method, we also propose a solution framework for an intelligent cooking assistant system for patients with Alzheimer's disease.

Chapter 6

Sequential Decision Making in Multi-Agent Systems

In complex multiagent systems, sequential decision making has long been a significant challenge with many characteristics. One prominent characteristic is the *interactions* among the agents. In many practical scenarios (e.g., the Stag-Hunt game), agents need to *cooperate* with each other to achieve a common goal with high rewards (e.g., hunting stags), while each individual agent is *self-interested* and might deviate from the cooperation for less risky goals with low rewards (e.g., hunting rabbits). Another critical characteristic comes from various *uncertainties*. One type of uncertainty arises from the lack of accurate knowledge of environment and other agents where the uncertain information can be modelled probabilistically. A more challenging type of uncertainty lies in the environment-related factors which we do not know how to model.

These two challenges are significantly amplified when it comes to *sequential decision making*, where we need to look at not only short term rewards but also rewards in the long run. Therefore, one has to consider subsequent effect of the current action, especially in a dynamic environment. Another crucial characteristic is the *limited learning trials*. On the Minecraft platform, it usually takes several seconds to complete one episode of game. Therefore, it is extremely time consuming to learn an effective policy.

The Microsoft Malmo Collaborative AI Challenge (MCAC), which is designed to encourage research relating to various problems in Collaborative AI, builds on a

Minecraft mini-game [55] called “Pig Chase”. Pig Chase is played on a 9×9 grid where agents can either work together to catch the pig and achieve high scores, or give up cooperation and achieve low scores. After playing certain episodes (e.g., 100) of games, the agent who achieves the highest average scores wins the challenge. Despite its simple rules, this game has all the above stated key characteristics. Though there are numerous papers studying sequential decision making in complex multiagent systems, they only address a subset of these characteristics. We hope to shed some light on solving this class of problems by presenting HogRider, a champion agent which won 2017 Microsoft Malmo Collaborative AI Challenge [56], with the following key contributions.

First, via enormous simulations, we provide thorough analysis of the Minecraft game, and identify its key challenges. Specifically, we discover several important aspects of the game that are not revealed by the MCAC rule, such as the pig’s uncertain behavior model, the uncertainty in observing the other agent’s actions, among others.

Second, we propose a novel agent type hypothesis approach for dealing uncertainties about the type of the other agent and the observation of the other agent’s actions. When forming ad hoc teams, it is important to identify the types of the collaborating agents. A typical approach is to maintain a belief of their type, and update it based on the actions of the other agents using Bayes theorem [162]. However, we discover that the actions of the other agents could be incorrectly observed. To this end, we derive two adaptations to the traditional Bayes theorem to cope with this observation uncertainty.

Third, we come up with a novel Q-learning framework to learn an optimal policy against each type of agent, with the following novelties. First, state-action abstraction is utilized to reduce the huge state-action space. Besides, in classical tabular Q-learning, Q-values are usually randomly initialized, which requires a particularly large number of episodes for training. While it takes several seconds to complete an episode on the Minecraft platform, it is extremely time consuming to train an effective policy. We propose a warm start approach to initialize the tabular Q-function, which utilizes a decision tree framework derived from human reasoning. Moreover, we show that when learning trials are limited, random exploration of potential optimal actions is inefficient

Figure 6.1: Setting of MCAC. The left hand side of the figure is the first person view of the game, while the right hand side of the figure is the symbolic global view.

and sometimes even harmful for searching optimal policies. Therefore, we propose an active- ϵ -greedy method to make better tradeoff between exploitation and exploration.

Last, we conduct extensive experiments to evaluate HogRider. We demonstrate that by considering uncertainty in agent action observation, HogRider outperforms existing agent type belief update methods. We also show improved performance of HogRider compared with traditional Q-learning methods. Challenge results show that HogRider outperforms all the other teams by a significant edge, in terms of both optimality and stability. Moreover, we show that HogRider performs even better than human players.

The remainder of the chapter is organized as follows. Section 6.1 describes the settings and rules of the MCAC. Section 6.2 presents the solution algorithm HogRider to the described challenge problem. Section 6.3 shows the experimental evaluations of HogRider. Section 6.4 summarizes the lessons learned from this work as well as major contributions.

6.1 Challenge Description and Analysis

We briefly introduce the challenge background and analyze its key characteristics.

Figure 6.2: Two types of situations where a pig is caught

6.1.1 Settings & Rules

MCAC builds on a mini-game played within a 9×9 grid on the Minecraft platform (Figure 6.1). The green grids represent the grass where agents can walk on, the yellow grids represent the blocks where agents cannot walk on or pass through, and the two black grids represent two exits. There are two agents and a pig (which can be viewed as part of the environment). The red arrow represents the agent (RED) that we are going to design, the blue arrow represents the agent (BLUE) that agent RED needs to play with, and the pink dot represents a pig which needs to be caught. The *legal movements* of the agents include: move forward (*MF*), turn left (*TL*) and turn right (*TR*). Agent BLUE is provided by MCAC, which is designed to be either a *Random* agent or a *Focused* agent in each episode of game. A *Random* agent takes any of the three legal movements with an equal probability of $1/3$, while a *Focused* agent always chases the pig through the shortest path using A* search algorithm. In each episode of game, there is a probability of .25 for agent BLUE to be *Random*, and a probability of .75 to be *Focused*. This type does not change within one episode of game.

The goal of the challenge is to achieve a high score in certain episodes (e.g., 100 or 500) of games. After an episode started, two agents take actions in turn. If the pig is caught, each agent receives 25 points. To catch a pig, the pig has to be completely surrounded by blocks and the agents. Figure 6.2a shows an example where a pig can be caught by a single agent, i.e., when the pig goes to the exit, and one agent goes to the

Figure 6.3: Probability distribution of the pig's positions after agent RED takes one step, where the grid in the center is the original position of the pig, and darker color means higher probability. In Figures 6.3a-6.3c, agent RED respectively waits for (a) no time (b) 1 second, and (c) 3 seconds before it takes a step. Note the pig does have a very short time (around 0.1 seconds) moving even if agent RED does not wait. This time is the response time of the Minecraft platform.

only grass grid beside the exit. Figure 6.2b shows an example where a pig can only be caught by two agents. An alternative is to give up cooperation and go to the exit. In this case, only the agent who *first* reaches the exit can get 5 points. Our agent loses 1 point whenever one step is made, and one episode of game terminates as long as one of the following conditions is met: 1) the pig is caught, 2) one agent reaches the exit, 3) 25 steps are taken, or 4) the maximum game time (about 100 seconds) is used up.

6.1.2 Analysis and Key Challenges

Uncertain behavior model of the pig. The behavior model of the pig is crucial to generating efficient policies, whereas it is not released by MCAC. We keep statistics of the pig's positions after our agent takes one step. After 10,000 steps, we draw the probability distribution of the pig's positions. As shown in Figure 6.3, we can see that 1) the pig does not follow the legal movements as human agents, and is able to move multiple steps or even make turns when human agent takes one step, 2) the probability of the pig moving to each direction is almost equal, and 3) with longer waiting time before our agent RED takes a step, the higher probability the pig moves. The statistics gives us a hint that it might be beneficial to wait for a few seconds when the pig is in a position that is uncachable, so that it might move to positions that are catchable.

Uncertainties in the type of agent BLUE & observation of its actions. In each episode of game, agent BLUE has a .25 and a .75 probability of being *Random* or

Focused, respectively. While correctly predicting its type relies on accurate observation of its movements, we noticed that the action observed (returned by the Minecraft simulator) has a probability of around .25 to be incorrect. It would be highly risky if this observation uncertainty is neglected.

Coexistence of cooperation & self-interest. To catch a pig (25 points) in situations like in Figure 6.2b, agent RED must cooperate with agent BLUE. Under some other situations, it might be more beneficial to deviate from the cooperation and move to the exit (5 points), especially when agent RED is close to the door, or there are few steps left.

Sequential decision making. Apart from the above challenges, our agent needs to sequentially make decisions during each episode of game. This significantly adds to the computational complexity of the problem.

Limited learning trials. The simulations are performed on the Minecraft platform, which usually takes several seconds for one episode of game. As a result, it is extremely time consuming to run a large number of simulations. For example, if one episode runs for 5 seconds, it would take around 6 days to train 100,000 episodes.¹

6.2 Solution Approach

In this section, we present key ideas behind HogRider. First, we design an agent type hypothesis framework to update the belief over the types of agent BLUE, which is robust to uncertainty in agent action observation. On top of that, we propose a novel Q-learning approach to learn an optimal policy against each type of agent BLUE.

6.2.1 Belief Update for Agent Types

Formally, we denote a set of agent types as $\mathcal{T} = \{T\} = \{Random, Focused\}$. We have a prior probability distribution over the agent types, i.e., $P(Random) =$

¹As we will introduce in the next section, this number is smaller than half of the state-action space size. An alternative to the Minecraft platform is to build our own simulator by formalizing a model for the game. However, the uncertainties discussed above cannot be accurately modelled.

0.25, $P(\text{Focused}) = 0.75$. At each position, we also know the conditional probability $P(a|T)$ of a type T agent taking movement $a \in \{MF, TL, TR\}$. According to the Bayes theorem, the posterior probability of an agent being type T when it takes movement a is:

$$P(T|a) = \frac{P(a|T) \cdot P(T)}{P(a)}, \quad (6.1)$$

where $P(a)$ is the probability of movement a being taken.

Generalizing Bayes theorem. As the movements of agent BLUE might be incorrectly observed with a probability of $P_u \approx .25$. We now derive the belief update rule which considers the uncertainty in observing agents' actions.

Theorem 6.1. *With the existence of uncertainty in observing the action of agent BLUE, the agent belief update rule is*

$$\begin{aligned} P(T|a) &= \frac{P(T) \cdot [(1-P_u) \cdot P(a|T) + P_u \cdot P(a|u)]}{(1-P_u) \cdot \sum_{T \in \mathcal{T}} P(a|T) \cdot P(T) + P_u \cdot P(a|u)} \\ &= P(T) \varphi(T, a) \quad (\text{for simplicity}) \end{aligned} \quad (6.2)$$

where $P(a|u)$ is the conditional probability of movement a being observed when the observation is incorrect.

Proof. With observation uncertainty, we rewrite $P(a|T)$ as

$$P'(a|T) = (1 - P_u) \cdot P(a|T) + P_u \cdot P(a|u). \quad (6.3)$$

Similarly, the probability $P(a)$ of an action being taken is reformulated as

$$P'(a) = (1 - P_u) \cdot \sum_{T \in \mathcal{T}} P(a|T) \cdot P(T) + P_u \cdot P(a|u). \quad (6.4)$$

Substituting Eqs.(6.3)-(6.4) to Eq.(6.1), we obtain Eq.(6.2). $\square$

Squash Bayes update with hyperbolic tangent function. Even with the generalization to Bayes theorem, a single wrong action observation could drastically drop the posterior probability of the correct agent type. To mitigate this issue, we propose a more conservative update method, which uses a shifted hyperbolic tangent function

$sq(x) = \tanh(x - 1) + 1 = \frac{e^{2(x-1)} - 1}{e^{2(x-1)} + 1} + 1$ to squash the factor $\varphi(T, a)$ to a value between 0.5 and 2, such that the agent type belief is updated to a new value within (0.5, 2) times of the original value. Formally, the Squash Bayes update function is:

$$P(T|a) = P(T)\varphi'(T, a), \quad (6.5)$$

where

$$\varphi'(T, a) = \begin{cases} sq(\varphi(T, a)), & \varphi(T, a) \geq 1 \\ \frac{1}{sq(1/\varphi(T, a))}, & \varphi(T, a) < 1 \end{cases}. \quad (6.6)$$

6.2.2 Adapt Q-Learning to Various Challenges

Q-learning [15] is a model-free reinforcement learning technique for finding optimal policies in sequential decision making problems. We adapt Q-learning to deal with various complexities in the problem. Formally, we formulate the Minecraft game as a Markov Decision Process (MDP). The *state* $s \in \mathcal{S}$ is defined as a four-dimension vector, which includes the position and facing direction of agent RED, the position and facing direction of agent BLUE, the position of the pig (the pig's behavior does not depend on its facing direction), as well as the steps that have already been taken. The action $A \in \mathcal{A}$ of the MDP is a tuple $A = \langle w, a \rangle$, where w is a binary variable indicating whether HogRider needs to wait for a certain time period (usually 3 seconds), and a is the legal movement after the wait option.¹ A *Q-function* maps a state action pair (s, A) into real values:

$$Q : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{R} \quad (6.7)$$

A *policy* $\pi(A|s)$ is a mapping from any state s to an action A that should be taken in that state. In Q-learning, an important challenge is to perform the “exploitation” of the current optimal action, while balancing the “exploration” of potentially better actions. For example, in the ε -greedy policy, it selects the action with the largest Q-value (i.e., $A^* = \arg \max Q(s, A)$) with probability $1 - \varepsilon$, and selects a random action with probability ε .

¹Note that action is different from the legal *movement* in the sense that it has one more dimension of variable w .

Usually, Q-learning starts with a randomized initialization of Q-function $Q(s, A)$. It then generates an *episode* (e.g., one round of game on the Minecraft platform) by performing actions derived from the policy, while updating Q-function with a *stochastic gradient descent* method, using the state-action values returned from the generated episode. Q-learning is guaranteed to converge after sufficient observations of each action at each state [15].

State-Action Abstraction. For each agent, the number of its feasible positions is $25 - 4 = 21$ and there are 4 directions for each position. For the pig, since it can stay at the two exits, the number of feasible positions is 23. Consequently, the possible combinations of the agents and the pig's positions are $21 \times 4 \times 21 \times 4 \times 23 = 162,288$. Taking the last dimension of state (i.e., the number of steps have already been taken) into consideration, the possible number of states is $162,288 \times 25 = 4,057,200$, which is a huge number considering the average running time for an episode. While playing the game, we notice that:

It is the relative positions (i.e., distances) between the agents, the pig and the exits, rather than the absolute positions, that determines a player's strategy.

We propose a state abstraction method based on the intuition. An abstracted state has five dimensions, i.e., *PigStat* which indicates whether the pig is uncachable ($PigStat = 0$), catchable by a single agent ($PigStat = 1$) or catchable by 2 agents ($PigStat = 2$), $d(R, P)$ which indicates the distance between agent RED and the pig, $d(B, P)$ which denotes the distance between agent BLUE and the pig, $d(R, E)$ which denotes the distance between agent RED and the nearest exit, and *step* which denotes the number of steps that have been taken. Since *PigStat* has three possible states, the distance of an agent to the pig ranges from 0 to 9, and the distance of agent RED to the nearest exit ranges from 0 to 7, the total number of states in the abstracted state representation is $3 \times 10 \times 10 \times 8 \times 25 = 60,000$, which is less than 15% of the original state space. To transform from the original representation of states to the new one, we only need to calculate the three types of distances based on the original state representation, the calculation overhead of which is trivial.

One issue raised from the abstracted state representation is that, in the new state space, we ignore the direction of agent RED, making it infeasible to directly decide its legal movement. To handle this issue, we propose a corresponding action representation, i.e., a binary variable w which is the same as the original action representation, and a binary variable a' which indicates whether to chase the pig or to head for the exit. To translate the new action representation to the original one, we only need to relate to the position of agent RED, the computation overhead of which is also negligible. Moreover, the action space is reduced from $2 \times 3 = 6$ to $2 \times 2 = 4$. With new representations of states and actions, we can reduce the size of the tabular Q-function to less than 1%, with only 240,000 state-action pairs.

Warm Start Using Human Reasoning. As analyzed above, even after abstraction, there are still 240,000 state-action pairs. In traditional Q-learning, where Q-function is usually arbitrarily initialized, it is required to run millions of episodes to ensure convergence, which takes days due to the long simulation time. Human demonstration [163] was proposed to improve efficiency of reinforcement learning, where humans directly assign values to initialize the Q-function. However, human demonstration becomes rather inefficient when there are too many state-action pairs. We take a further step by utilizing a human reasoning-derived decision tree for Q-function's warm start.

The key principles of the warm start approach include 1) when making instant decisions (i.e., $w=0$), we ignore the pig's movement according to Figure 6.3a; 2) we wait for the pig to move when it is at an uncatchable position; and 3) we always choose the action with the highest estimated score. Figure 6.4 shows the flow of the decision tree for playing with the *Focused* agent. It starts with generating state $s = \langle PigStat, d(B, P), d(R, P), d(R, E), step \rangle$. We first check the value of *PigStat*. If *PigStat* = 1, we further check distance $d(R, P)$. Then we initialize the Q values of different actions accordingly. For simplicity, we use $w = 0$ (or 1) to represent an action without (or with) waiting time and $a' = 0$ (or 1) to represent chasing the pig (or leaving the game) for $A = \langle w, a' \rangle$. For state s , we can compute the probability $N_m(0)$ (and $N_m(1)$) that pig stays unmoved with 25-step steps remaining for $w = 0$ (and $w = 1$). $score_{exit}$ is what agent RED can get by leaving the game through exits. $Min_Estimate$ is the estimated score when the pig stays unmoved with probability

Figure 6.4: Flow of the human reasoning-derived decision tree

$N_m(0)$ for $PigStat = 0$ (or moves away with probability $1 - N_m(1)$ for $PigStat > 0$); $Max_Estimate$ is the score for agent RED when the pig is caught at its current position (or the nearest catchable position). We initialize the Q-function for playing with *Random* agent in a similar way.

Active- ϵ -Greedy. To explore a potentially better action, classical tabular Q-learning methods usually explore the entire action space (e.g., ϵ -greedy). Counter-intuitively, experimental results (to be explained later in Figure 6.6a) show that the performance of ϵ -greedy (the line with solid circle markers) declines in the early stage of learning. This is because the optimal actions estimated by our constructed decision tree are usually very good. Therefore, exploring suboptimal actions would lead to performance deterioration in a short term. Meanwhile, if we always deterministically exploit an action that has the highest Q-value, there would be very limited improvement to the current policy. To this end, we propose an *active- ϵ -greedy* approach to balance the exploration and exploitation. In active- ϵ -greedy, we still select the optimal action based on the Q-function with the probability of $1 - \epsilon$. Different from ϵ -greedy, we explore sub-optimal actions with probability ϵ , when the Q-value of that action is no less than certain percentage (hand tuned as 50% in practice) of the highest Q-value in the same state.

Training HogRider. In the training process, the type of agent BLUE is fixed as

either *Random* or *Focused*. As shown in Algorithm 11, training starts with initialization via the decision tree derived from human reasoning. It then enters the outer loop (Lines 2-8). In each iteration (episode) of the outer loop, HogRider first initializes the state s (Line 3). It then enters the inner loop (Lines 4-7). In each iteration (step) of the inner loop, HogRider first chooses an action A according to the active- ε -greedy method (Line 5). It then takes the action (after that, agent BLUE will alternatively take an action) and observes the immediate reward R and the next state s' (Line 6), which will be assigned as the starting state of the next step (Line 7). The inner loop terminates when a terminal state is reached. After that, Q-values for all the state-action pairs (s, A) that have been visited during this episode are updated (Line 8), where α is the learning rate. $V(s, A) = \sum_{s'=s}^{terminal} R(s', A)$ denotes the “real” value of the state-action pair (s, A) , which is the sum of immediate reward of all the state action pairs that are visited after state s during the episode. The entire outer loop terminates after M (e.g., 3,000) episodes.

Algorithm 11: Training HogRider

```

1  $Q(s, A) \leftarrow$  human reasoning,  $\forall s, A$ ;
2 while #episodes  $\leq M$  do
3   Initialize state  $s$ ;
4   while  $s$  is not terminal do
5     Choose  $A$  with active- $\varepsilon$ -greedy method;
6     Take action  $A$ , observe  $R(s, A)$  and  $s'$ ;
7      $s \leftarrow s'$ ;
8    $Q(s, A) \leftarrow (1 - \alpha)Q(s, A) + \alpha V(s, A)$ ,  $\forall$  visited  $(s, A)$ 
9 return  $Q(s, A)$ ,  $\forall s, A$ ;
```

6.2.3 HogRider

After training, we obtain Q-functions for both agents *Random* and *Focused*, which are denoted as Q_R and Q_F , respectively. We now present HogRider (Algorithm 12) which integrates the learned type-specific Q-function with the agent type hypothesis framework. Note that unlike the training process, agent BLUE can be either *Random* or *Focused* for different episodes of games. At the beginning of each episode of game, HogRider initializes the belief of agent BLUE being *Random* as $P(Random) = 0.25$

(Line 1). At the same time, HogRider observes the initial state s of the game and takes the first action according to the initial belief (Line 3). After that, it observes the new state s' (Line 4) and enters the while loop (Lines 5-9).

Algorithm 12: HogRider

```

1  $P(Random) \leftarrow 0.25$ ;
2 Observe initial state  $s$ ;
3 Take action  $A$  w.r.t.  $s$  and  $P(Random)$ ;
4 Observe  $s'$ ;
5 while  $s'$  is not terminal do
6   Observe action of agent BLUE;
7   Update  $P(Random)$  using Eq.(6.5);
8   Take action  $A$  w.r.t.  $s'$  and  $P(Random)$ ;
9    $s \leftarrow s'$  and observe new state  $s'$ ;
```

Different strategies of selecting an optimal action. In each iteration, it observes the action of agent BLUE in last step (Line 6) according to the difference between s and s' , updates the belief accordingly (Line 7), takes an action A based on the current state s' and $P(Random)$ (Line 8). There are three ways to select an action. One way (called *Separate*) is to compare $P(Random)$ with a threshold P_0 , and choose $A^* = \arg \max Q_R(s, A)$ when $P(Random) \geq P_0$ (vice versa). The second way (called *Weighted*) is to maintain an expected Q-value for each action A : $\bar{Q}(s, A) = P(Random) \cdot Q_R(s, A) + P(Focused) \cdot Q_F(s, A)$, and select $A^* = \arg \max \bar{Q}_R(s, A)$. These two methods are deterministic. The last way (called *Mixed*) is to use $A^* = \arg \max Q_R(s, A)$ with a probability $P(Random)$, and use $A^* = \arg \max Q_F(s, A)$ with a probability of $P(Focused)$, which is non-deterministic. After taking an action, it stores the current state in s and observes a new state s' (Line 9). This process until s' is a terminal state.

6.3 Experimental Evaluations

In this section, we present the challenge results of the top 5 teams in MCAC 2017. We also conduct extensive experiments to evaluate performance of various ideas from the proposed solution algorithm.¹

¹There are two ways to measure the performance of different methods, i.e., per step score and per game (episode) score. We only show results for per step scores, while we refer to the Appendix in

6.3.0.1 MCAC Results

Table 6.1 shows the performance of the top 5 teams in MCAC 2017. In terms of both per step mean score and variance/mean, HogRider achieves the best performance. Compared to the second best team, HogRider wins with an edge of 13% in mean score, and an edge of 21.7% in variance/mean. This demonstrates that HogRider outperforms other teams in both optimality and stability.

Team	Mean ($\uparrow$)	Variance	Variance/Mean ($\downarrow$)
HogRider	2.752	2.55	0.9266
The Danish Puppeteers	2.435	2.88	1.1828
Bacon Gulch	1.681	2.18	1.2968
Village People	1.618	2.05	1.2670
AASMA	0.591	1.07	1.8105

TABLE 6.1: Performance of different teams in MCAC. $\uparrow$ means the higher, the better, while $\downarrow$ means the opposite. The best performance is highlighted for all the tables.

6.3.0.2 Strategies for Selecting the Optimal Action

As we described, there are three strategies to select an optimal action based on the Q-function, i.e., *Separate*, *Weighted* and *Mixed*. Table 6.2 shows the performance of the three different strategies, where *Separate* has the best performance in both optimality and stability. In the following experiments, we utilize *Separate* as the base strategy for optimal action selection.

	Mean score ($\uparrow$)	Variance	Variance/Mean ($\downarrow$)
Separate	2.752	2.55	0.9266
Weighted	2.527	2.36	0.9339
Mixed	2.478	2.81	1.1340

TABLE 6.2: Strategies for selecting the optimal action

Section 6.5 for results of per episode scores, as well as figures showing the detailed curves of the results that cannot be presented with tables.

Figure 6.5: Evaluate human reasoning-aided initialization

6.3.0.3 Belief Update

Table 6.3 evaluates the performance of HogRider using different methods for updating the beliefs over agent types, where Bayes means traditional Bayes theorem in Eq.(6.1), G-Bayes means the extension of Bayes theorem in Eq.(6.2), and G-Bayes+tanh means G-Bayes with a hyperbolic tangent squashing function in Eq.(6.5).

Method	Accuracy	Mean($\uparrow$)	Var.	Var./Mean($\downarrow$)
Bayes	0.578, 0.955 (0.672)	2.456	2.45	0.9976
G-Bayes	0.701, 0.886 (0.747)	2.505	2.27	0.9062
G-Bayes+tanh	0.961, 0.863 (0.937)	2.752	2.55	0.9266

TABLE 6.3: Different methods for agent type belief update

Note that for the Accuracy column, the three numbers are respectively the detection accuracies of two agent BLUE types and the expected overall accuracy. In general, G-Bayes+tanh has the highest overall detection accuracy and optimality among all methods, while being slightly inferior to G-Bayes in stability. Compared with the second best approach, G-Bayes+tanh is better with an edge of 9.9% in optimality, and an edge of 25% in overall detection accuracy.

6.3.0.4 Warm Start

Figure 6.5 shows the learning curve and validation curve of HogRider with human reasoning-aided initialization playing with the *Focused* agent. The validation curve

Figure 6.6: Evaluate different exploration methods

shows that, using initialization, the mean score of HogRider significantly improves by 10.9%, while variance/mean decreases by 5.7%. The learning curve indicates that we can learn a better Q-function much faster.

6.3.0.5 Different Exploration Methods

Figure 6.6 compares the learning curve and validation curve of the active- ϵ -greedy with other exploration methods (playing with *Focused* agent), where Deterministic always selects the current optimal action $A^* = \arg \max Q(s, A)$, ϵ -greedy selects A^* with probability $1 - \epsilon$ and randomly explores other actions with probability ϵ , Semi-Deterministic selects A^* with probability $1 - \epsilon$ and explores the second best action with probability ϵ . From the learning curve, we can see that 1) arbitrary exploration (i.e., ϵ -greedy) can be rather harmful when learning trials are limited, 2) deterministic methods provide limited help in improving the current policy, 3) semi-deterministic methods always explore the second-best action without considering the estimated value of it, which is also inefficient, and 4) with active- ϵ -greedy, the potential optimal actions (with at least 50% estimated value of the best action) are exploited and the policy keeps improving along the training process. The validation curve demonstrates the effectiveness of active- ϵ -greedy method. In terms of mean score, active- ϵ -greedy is better with an edge of 4.1% compared with the second best approach (Deterministic), and better with an edge of 6.7% compared with ϵ -greedy.

6.3.0.6 Compare HogRider with Human Players

We also invited 10 human players from our university to play the Pig Chase game, all of whom are PhD candidates. Surprisingly, the results show that HogRider performs even better than human players with an edge of 28.1% in mean increase and 29.6% in variance/mean decrease.

Approach	Mean($\uparrow$)	Variance	Variance/Mean($\downarrow$)
HogRider	2.752	2.55	0.9266
Human	2.149	2.83	1.3169

TABLE 6.4: Comparison with human players

6.4 Chapter Summary

6.4.1 Lessons Learned

Developing HogRider has been a progressive experience. Different from the constructive flow that presents HogRider, it was rather a step by step process where new discoveries and ideas interchangeably emerge. We gained several insights into solving challenging problems in the areas of collaborative AI, as well as other more general research domains. **First, “if you know yourself and your enemy, you will never lose a battle”.**¹ It has been a long exploration before we set up the integrated framework of agent type hypothesis and Q-learning. Our initial plan was to simply learn an effective policy with Q-learning without differentiating different types of agents. Despite the model-free property of Q-learning, its direct application worked poorly simply because we did not exploit the problem structure. While looking for cutting-edge tools is important, learning the foundations of the problem ensures the right direction. **Second, human intuition can take machines to the heights.** Due to its popularity, we initially decided to use DQN [164] as the learning approach to learn policies against each agent type. Despite its power in expressing Q-functions, initialization does not work due to the parameterized Q-functions, which poses DQN at a disadvantageous

¹It is from an ancient Chinese military treatise, *The Art of War*.

place against tabular Q-learning approaches. In fact, a few teams using DQN had poor performance in the challenge. It is rather surprising and enlightening how much human reasoning could help. This type of initialization could be utilized in other areas where domain knowledge can help derive human reasoning. **Moreover, models and solution algorithms should be kept updating with new discoveries of latent properties.** For example, the uncertainty in agent action observation was not discovered until the algorithm was almost built. To this end, we made two additional adaptations to the traditional Bayes theorem which brought significant improvement to the algorithm.

6.4.2 Summary of Contributions

There are many characteristics such as complex interactions among agents, uncertainties, and limited learning trials that make effective sequential planning extremely challenging. The MCAC, which is designed to encourage research relating to various problems in Collaborative AI, is a typical test bed with all these challenges. We introduce HogRider, champion agent of MCAC in 2017, which has two key innovations. First, we design a generalized agent type hypothesis framework to handle the uncertainties in agent type and observation in the other agent's action. Second, we propose a novel Q-learning approach to learn effective policies against each type of collaborating agents with various novel ideas, including state-action abstraction, a human reasoning-aided approach to initialize Q-function, and an active greedy strategy to balance the exploitation and exploration.

6.5 Appendix

Since MCAC only evaluate the competition results according to the per step mean score and variance, we also only show corresponding results in the main text. We provide more experimental results on per episode scores in the appendix.

Strategies for selecting the Optimal Action

Table 6.5 shows the per episode results of different strategies. As we can see, Separate is the best one in terms of both optimality and stability.

Approach	Mean($\uparrow$)	Variance	Variance/Mean($\downarrow$)
Separate	14.414	8.72	0.6050
Weighted	13.116	9.53	0.7266
Mixed	12.636	9.64	0.7629

TABLE 6.5: Per episode results for different strategies

Figure 6.7 shows the per episode and step mean score curve of different strategies.

Figure 6.7: Mean score curve for 3 strategies

Belief Update

Table 6.6 shows the per episode results for different belief update approaches. G-Bayes+tanh is better with an edge of 4.8% increase in mean and 7.7% decrease in variance/mean comparing to G-Bayes.

Approach	Mean($\uparrow$)	Variance	Variance/Mean($\downarrow$)
Bayes	12.794	9.74	0.7613
G-Bayes	13.186	9.55	0.7243
G-Bayes+tanh	14.414	8.72	0.6050

TABLE 6.6: Per episode results for different belief update approaches

Figure 6.8 shows the accumulated detection accuracy of different belief update methods, where the accumulated accuracy value for step x is the overall detection accuracy for all steps $\leq x$. As we can see, G-Bayes-tanh performs best for *Focused* agent detection, while its detection accuracy for the *Random* agent is slightly worse than the other two methods. Considering the probability that agent BLUE to be *Focused* (.75) or *Random* (.25), G-Bayes+tanh has a best overall accuracy, which is 93.7% and it is 25.4% higher than the second best G-Bayes update method.

Figure 6.8: Accumulated detection accuracy

Warm Start

Figure 6.9: Per episode results for Q-learning with warm start

Figure 6.9 shows the learning curve and validation curve for Q-learning per episode results with/without warm start while playing with *Focused* agent. As we can see, with

Figure 6.10: Per episode results for comparison among 4 exploration methods

warm start, the Q-learning performs much better. Specifically, Table 6.7 shows the per episode results for Q-learning results while playing with *Focused* agent with/without warm start. With human-reasoning aided initialization, the mean score is improved by 7.3%, meanwhile the variance/mean is decreased by 39.4%. Table 6.8 shows the per step results. With human-reasoning aided initialization, the mean score is improved by 10.9%, meanwhile the variance/mean is decreased by 5.7%.

Approach	Mean($\uparrow$)	Variance	Variance/Mean($\downarrow$)
Warm	17.862	4.61	0.2581
No Warm	16.653	7.09	0.4257

TABLE 6.7: Per episode results for warm start

Approach	Mean($\uparrow$)	Variance	Variance/Mean($\downarrow$)
Warm	3.284	2.28	0.6943
No Warm	2.960	2.18	0.7365

TABLE 6.8: Per step results for warm start

Different Exploration Methods

Figure 6.10 shows the per episode mean score learning curve and validation curve for different exploration methods while playing with *Focused* agent. As we can see, Active- ϵ -greedy method learns the best Q-function and outperforms other exploration methods.

Approach	Mean($\uparrow$)	Var.	Var./Mean($\downarrow$)
Deterministic	17.215	6.07	0.3526
ϵ -greedy	17.346	5.17	0.2981
Semi-Deterministic	17.344	5.47	0.3154
Active- ϵ -greedy	17.862	4.61	0.2581

TABLE 6.9: Per episode results for exploration methods

Table 6.9 shows the per episode results for different exploration methods. Active- ϵ -greedy method has 3% increase in mean score comparing to ϵ -greedy, meanwhile 13.4% decrease in variance/mean. Table 6.10 shows the per step results for different exploration methods.

Approach	Mean($\uparrow$)	Var.	Var./Mean($\downarrow$)
Deterministic	3.1543	2.17	0.6880
ϵ -greedy	3.0778	1.93	0.6270
Semi-Deterministic	3.1464	2.26	0.7184
Active- ϵ -greedy	3.284	2.28	0.6943

TABLE 6.10: Per step results for exploration methods

Compare HogRider with Human Players

Table 6.11 provides the per episode results for HogRider in comparison with human players. HogRider increase the mean score by 36.8% and decrease the variance/mean value by 48.2%.

Approach	Mean($\uparrow$)	Variance	Variance/Mean($\downarrow$)
HogRider	14.414	8.72	0.6050
Human	10.540	12.30	1.1670

TABLE 6.11: Per episode results V.S. human players

Chapter 7

Conclusion and Future Work

This thesis investigates several important problems in the area of large scale decision making in multiagent systems. In this chapter, we will summarize the major contributions in this thesis and discuss several potential future research directions.

7.1 Conclusion

This thesis makes a number of contributions to both theoretical and practical aspects of the state of the art of large scale strategic decision making in multiagent systems.

We start with a data-driven decision making in the airlines industry, and propose the MAP framework for optimal airline flight frequency allocation over multiple routes. We make three key contributions. i) We design a novel ensemble predictor based on past regression-based work, clustering techniques and game theoretic analysis, which have never been utilized for this purpose; ii) based on the prediction, we design several algorithms to solve the profit maximization problem; iii) we conduct extensive experiments. We show that the prediction performance of MAP-EF is much better than past methods. We also compare runtime and optimality of our proposed optimization algorithms with benchmarks. We also show that MAP increases profitability per route by at least 55%. Even if there are factors we did not consider on the basis of open source data and these

numbers are reduced substantially if private airline data was to be used, the increased profitability would still be substantial.

We extend the decision making from the single agent scenario to multiagent systems, with two applications to the airlines and cloud computing domains. We first extend the airline frequency allocation problem by considering multiple competing airlines over a same set of routes, and propose ITER-Freq algorithm to compute the Nash equilibrium solution. We then look into a more complicated vertical and horizontal cloud federation scenario, where both competition and cooperation exist amongst a group of agents. With this work, we make the first attempt to reveal the interrelated workload factoring and coalition formation among secondary clouds, and answer the following intrinsic questions: 1) What is the most stable coalition structure and payoff division; 2) What are the optimal (equilibrium) workload factoring strategies of different coalitions; 3) Is it beneficial for the secondary clouds to join such a VHCF network? The experimental results show that interestingly, the two common practices in the real world - no-cooperation and the grand coalition - are sometimes not the most stable cooperation pattern. Instead, the most stable coalition structure usually lies between these two extreme cases. Furthermore, the experimental results show that by implementing the VHCF network, the service delay penalty of the secondary clouds decreases by around 11% compared with the two common practices. We also look into a large scale sequential decision making problem in an environment with dynamism and uncertainties, which is applied in the ETC domain. We make three key contributions to this problem. First, we propose a formal model of the DyETC problem, which is formulated as a discrete-time MDP. Second, we develop a novel solution algorithm, PG- β to solve the formulated large scale MDP. Third, we conduct extensive experimental evaluations to compare our proposed method with existing tolling schemes. The results show that on a real world traffic network in Singapore, PG- β increases the traffic volume by around 8%, and reduces the travel time by around 14.6% during rush hour.

Finally, we investigate a more complicated scenario where agents need to do a sequence of decisions in a multiagent system. here are many characteristics such as complex interactions among agents, uncertainties, and limited learning trials that make effective sequential planning extremely challenging. This work leads to the design of our

agent HogRider, which won the Microsoft Malmo Collaborative AI Challenge 2017. HogRider has two key innovations. First, we design a generalized agent type hypothesis framework to handle the uncertainties in agent type and observation in the other agent's action. Second, we propose a novel Q-learning approach to learn effective policies against each type of collaborating agents with various novel ideas, including state-action abstraction, a human reasoning-aided approach to initialize Q-function, and an active greedy strategy to balance the exploitation and exploration.

7.2 Future Work

We have proposed novel solution techniques to a number of challenging and practical problems in the area of large scale strategic decision making in multiagent systems. In this section, we discuss several potential future research directions.

7.2.1 Various Extensions

For large scale data-driven decision making in a single agent scenario, we may extend our framework to problems with multiple decision variables. For example, in the airline profit maximization problem, we may consider joint-optimization over frequency and pricing. However, the monotonicity property of revenue and cost is not guaranteed with ticket pricing¹ and it is already one of the hardest KPs and RAPs with only frequency being considered. Moreover, we can extend one-shot frequency allocation problem to sequential decision making, where the current frequency allocation strategies will not only affect the immediate profit, but also future profits.

For sequential decision making, while our current work focused on a single domain with a relatively small scale traffic network, we can extend DyETC to larger scale networks. Potential approaches include stochastic optimization methods such as CMA-ES [165], continuous control variants of DQN [93] such as DDPG [166] and

¹By decreasing ticket price, market share will increase but revenue may or may not increase due to the change of price.

NAF [167], parallel reinforcement learning such as A3C [168] and variance reduction gradient methods such as Averaged-DQN [169].

For sequential decision making in a multiagent system, one potential future research is to look at scenarios where agents are completely unknown to each other before they meet. For example, there are no prior knowledge of the distribution of agent types, or there are no pre-defined types for the collaborating agents. Another potential research is to propose solution algorithms which can be generalized to different types of environments. For example, if the positions of the blocks are replaced, more blocks are added, or the size of grids are varied, how could the policy learned from the old setting applied to the new setting? In these scenarios, algorithms might have to combine off-line trained policies with online learning [170], which resembles the idea of Endgame solving [171] in Computer Poker games, and the set of transfer learning algorithms that are designed for reinforcement learning domains [172].

For all the frameworks developed in this thesis, they could be adapted to a variety of application domains with similar characteristics. For example, the MAP framework could be adapted to all types of budget allocation problems where a company could integrate data mining with optimization techniques. The VHCF framework could be applied to any domain (e.g., e-commerce, games, etc.) where there are both competition and cooperation among a group of agents. The DyETC framework could be extended to other sequential decision making problems based on a road network or any other graphs. Finally, the Malmo framework could be adapted to situations with uncertainty, dynamism and multiple players. One straightforward extension is to design AI agents in E-sports games such as WarcraftIII and Dota2.

7.2.2 Applying Large Scale Decision Making Techniques to Healthcare Problems

As one of the major motivations in doing the research relating to large scale decision making in multiagent systems, the applications to healthcare related problems are of particular interests.

Intelligent home care assistance to elderly people with Alzheimer's disease. In 2016, it is estimated that nearly 44 million people have Alzheimer's or a related dementia worldwide, and the global cost of caring Alzheimer's and dementia is estimated to be 605 billion, which is equivalent to 1% of the entire world's gross domestic product [158], and the situation is growing worse over time. In this respect, we hope to apply the techniques proposed in this thesis to develop intelligent decision making systems, which can assist the daily care of the elderly people in situations such as hospital care and home care. Such intelligent assistance could be, reminders to the patients of taking medicine regularly, advice to the patients on performing certain tasks (e.g., cooking, washing hands), etc. However, building such an intelligent agent is non-trivial in terms of technical challenges. Taking home care as an example, the status of an elderly people usually consists of several aspects, e.g., blood pressure, body temperature, activity, and so on. Given the huge number of status dimensions, the state space of the decision making process would be huge.

Another challenge in constructing such an elderly care intelligent system is that, the habit of a specific elderly people is usually unknown to the system *a priori*. As a result, it is necessary to build a framework similar to MAP, which integrates data-driven techniques with large scale decision making. At the early stage of the system, data could be collected to learn the personal habits of the target patient, e.g., the regular time when the patient gets up or goes to bed, the indications of the patient's facial expressions, etc. Later, optimization techniques could be applied to recommend the optimal activities to the patient, given a set of candidate activities. Moreover, the activities of the elderly people are usually sequential instead of one-shot, which makes an MDP-based approach close fit to the problem domain.

Healthcare resource allocation. Resource allocation problems in the healthcare domain are ubiquitous – from a microscopic level of allocating a doctor's slots to different patients, to a macroscopic level of allocating all types of resources (e.g., blood, surgery equipments) among different hospitals.

The decision making in healthcare is inherently challenging in terms of scalability – it could involve a large number of hospitals, various types of resources, and huge numbers of patients. The solutions developed in solving the Knapsack problems and

resource allocations could be well adopted in such type of problems, while it remains challenging to scale up the techniques to problems which have so many factors to take into account. In the meantime, it is naturally a sequence of decisions – the decisions need to be made on a timely basis, and usually have long term effects instead of just short term impact. For example, in transferring blood resource from one hospital to another, the former needs to take into account the expected usage of blood resource in the future, and whether this transfer would leave itself in shortage of blood resources.

Moreover, such decision making usually involves multiple entities instead of a single entity. For example, the blood resource allocation may be possible from one private hospital to another, while both hospitals might be profit-driven and a centralized solution is practically infeasible. In such cases, it would be more meaningful to design mechanisms such as that of VHCF, which takes into consideration both efficiency of cooperation aspects as well as different entities' own interests from the competition aspects.

Appendix A

List of Publications

1. Bo An, Haipeng Chen, Noseong Park, and VS Subrahmanian. Map: Frequency-based maximization of airline profits based on an ensemble forecasting approach. In *KDD*, pages 421–430, 2016
2. Bo An, Haipeng Chen, Noseong Park, and VS Subrahmanian. Data-driven frequency-based airline profit maximization. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(4):61, 2017
3. Haipeng Chen, Bo An, Dusit Niyato, Yeng Chai Soh, and Chuanyan Miao. Workload factoring and resource sharing via joint vertical and horizontal cloud federation networks. *IEEE Journal on Selected Areas in Communications*, 35(3): 557–570, 2017
4. Haipeng Chen, Bo An, Guni Sharon, Josiah P Hanna, Peter Stone, Chunyan Miao, and Yeng Chai Soh. Dyetc: Dynamic electronic toll collection for traffic congestion alleviation. In *AAAI*, 2018
5. Yanhai Xiong, Haipeng Chen, Mengchen Zhao, and Bo An. Hogrider: Champion agent of microsoft malmo collaborative ai challenge. In *AAAI*, 2018

Bibliography

- [1] Yanhai Xiong, Haipeng Chen, Mengchen Zhao, and Bo An. Hogrider appendix. 2017. <http://AAAIMalmo.weebly.com>.
- [2] Bo An, Haipeng Chen, Noseong Park, and VS Subrahmanian. Map: Frequency-based maximization of airline profits based on an ensemble forecasting approach. In *KDD*, pages 421–430, 2016.
- [3] Bo An, Haipeng Chen, Noseong Park, and VS Subrahmanian. Data-driven frequency-based airline profit maximization. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(4):61, 2017.
- [4] Kenneth J Arrow and Gerard Debreu. Existence of an equilibrium for a competitive economy. *Econometrica: Journal of the Econometric Society*, pages 265–290, 1954.
- [5] Daniel Gabay and Herve Moulin. *On the uniqueness and stability of Nash’s equilibrium in non cooperative games*. Université Paris IX-Dauphine, Centre de recherche de mathématiques de la décision, 1978.
- [6] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani. *Algorithmic game theory*. Cambridge University Press, 2007.
- [7] Barry J Nalebuff, Adam Brandenburger, and Agus Maulana. *Co-opetition*. HarperCollinsBusiness London, 1996.
- [8] Maria Bengtsson and Sören Kock. Coopetition in business networks to cooperate and compete simultaneously. *Industrial marketing management*, 29(5):411–426, 2000.

- [9] Giovanni Battista Dagnino. Coopetition strategy: a new kind of interfirm dynamics for value creation. In *Coopetition strategy*, pages 45–63. Routledge, 2009.
- [10] Devi R Gnyawali and Byung-Jin Robert Park. Co-opetition and technological innovation in small and medium-sized enterprises: A multilevel conceptual model. *Journal of small business management*, 47(3):308–330, 2009.
- [11] Xueming Luo, Rebecca J Slotegraaf, and Xing Pan. Cross-functional coopetition: The simultaneous role of cooperation and competition within firms. *Journal of Marketing*, 70(2):67–80, 2006.
- [12] Anton Proskurnikov, Alexey Matveev, and Ming Cao. Consensus and polarization in altafini’s model with bidirectional time-varying network topologies. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pages 2112–2117. IEEE, 2014.
- [13] Jiangping Hu and Wei Xing Zheng. Emergent collective behaviors on coopetition networks. *Physics Letters A*, 378(26-27):1787–1796, 2014.
- [14] Richard Bellman. *Dynamic programming*. Courier Corporation, 2013.
- [15] Christopher John Cornish Hellaby Watkins. *Learning from Delayed Rewards*. PhD thesis, University of Cambridge England, 1989.
- [16] Andrew W Moore and Christopher G Atkeson. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, 13(1):103–130, 1993.
- [17] Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *ICCG*, pages 72–83, 2006.
- [18] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *ECML*, pages 282–293, 2006.
- [19] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

- [20] Milind Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7:83–124, 1997.
- [21] Barbara J Grosz and Sarit Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86(2):269–357, 1996.
- [22] Bryan Horling, Victor Lesser, Regis Vincent, Tom Wagner, Anita Raja, Shelley Zhang, Keith Decker, and Alan Garvey. The TAEMS white paper, 1999.
- [23] Satinder Singh, Michael Kearns, and Yishay Mansour. Nash convergence of gradient dynamics in general-sum games. In *UAI*, pages 541–548, 2000.
- [24] Vincent Conitzer and Tuomas Sandholm. Awesome: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. *Machine Learning*, 67(1-2):23–43, 2007.
- [25] Jakob Foerster, Yannis M Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *NIPS*, pages 2137–2145, 2016.
- [26] Paul E Dunne, Sarit Kraus, Efrat Manisterski, and Michael Wooldridge. Solving coalitional resource games. *Artificial Intelligence*, 174(1):20–50, 2010.
- [27] Enrique Munoz De Cote, Archie C Chapman, Adam M Sykulski, and Nicholas R Jennings. Automated planning in repeated adversarial games. *arXiv preprint arXiv:1203.3498*, 2012.
- [28] Rajkumar Buyya, Rajiv Ranjan, and Rodrigo N Calheiros. Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services. In *Algorithms and architectures for parallel processing*, pages 13–31. Springer, 2010.
- [29] Amir Nahir, Ariel Orda, and Danny Raz. Workload factoring with the cloud: A game-theoretic perspective. In *INFOCOM*, pages 2566–2570, 2012.
- [30] David Villegas, Norman Bobroff, Ivan Roderio, Javier Delgado, Yanbin Liu, Aditya Devarakonda, Liana Fong, S Masoud Sadjadi, and Manish Parashar.

- Cloud federation in a layered service model. *Journal of Computer and System Sciences*, 78(5):1330–1344, 2012.
- [31] Bill Steele. Aristotle: A federated cloud for academic research. <http://www.news.cornell.edu/stories/2015/11/aristotle-federated-cloud-academic-research>, 2015.
- [32] Eubrazil cloud connect. <http://www.eubrazilcloudconnect.eu/>.
- [33] Michael P Papazoglou. Cloud blueprints for integrating and managing cloud federations. In *Software Service and Application Engineering*, pages 102–119. Springer, 2012.
- [34] Francesco Tusa, Antonio Celesti, Maurizio Paone, Massimo Villari, and Antonio Puliafito. How clever-based clouds conceive horizontal and vertical federations. In *ISCC*, pages 167–172, 2011.
- [35] Di Niu and Baochun Li. An efficient distributed algorithm for resource allocation in large-scale coupled systems. In *INFOCOM*, pages 1501–1509, 2013.
- [36] Wei Wang, Baochun Li, and Ben Liang. Dominant resource fairness in cloud computing systems with heterogeneous servers. In *INFOCOM*, pages 583–591, 2014.
- [37] Hongxing Li, Chuan Wu, Zongpeng Li, and Francis CM Lau. Profit-maximizing virtual machine trading in a federation of selfish clouds. In *INFOCOM*, pages 25–29, 2013.
- [38] Liang Zhang, Weijie Wu, and Dan Wang. Time dependent pricing in wireless data networks: Flat-rate vs. usage-based schemes. In *INFOCOM*, pages 700–708, 2014.
- [39] Ruben Van den Bossche, Kurt Vanmechelen, and Jan Broeckhove. Cost-optimal scheduling in hybrid iaas clouds for deadline constrained workloads. In *CLOUD*, pages 228–235, 2010.

- [40] Wei-Jen Wang, Yue-Shan Chang, Win-Tsung Lo, and Yi-Kang Lee. Adaptive scheduling for parallel tasks with qos satisfaction for hybrid cloud environments. *The Journal of Supercomputing*, 66(2):783–811, 2013.
- [41] Yuan Feng, Baochun Li, and Bo Li. Price competition in an oligopoly market with multiple iaas cloud providers. *IEEE Transactions on Computers*, 63(1): 59–73, 2014.
- [42] Francesco Palmieri, Luigi Buonanno, Salvatore Venticinque, Rocco Aversa, and Beniamino Di Martino. A distributed scheduling framework based on selfish autonomous agents for federated cloud environments. *Future Generation Computer Systems*, 29(6):1461–1472, 2013.
- [43] Dusit Niyato, Athanasios V Vasilakos, and Zhu Kun. Resource and revenue sharing with coalition formation of cloud providers: Game theoretic approach. In *CCGrid*, pages 215–224, 2011.
- [44] Lena Mashayekhy and Daniel Grosu. A coalitional game-based mechanism for forming cloud federations. In *UCC*, pages 223–227, 2012.
- [45] Marco Guazzone, Cosimo Anglano, and Matteo Sereno. A game-theoretic approach to coalition formation in green cloud federations. In *CCGrid*, pages 618–625, 2014.
- [46] Sudarshan Guruacharya, Dusit Niyato, Mehdi Bennis, and Dong In Kim. Dynamic coalition formation for network mimo in small cell networks. *IEEE Transactions on Wireless Communications*, 12(10):5360–5372, 2013.
- [47] Tianyu Wang, Lingyang Song, Zhu Han, and Walid Saad. Distributed cooperative sensing in cognitive radio networks: An overlapping coalition formation approach. *IEEE Transactions on Communications*, 62(9):3144–3160, 2014.
- [48] Yong Xiao, Dusit Niyato, Zhu Han, and Kwang-Cheng Chen. Secondary users entering the pool: A joint optimization framework for spectrum pooling. *IEEE Journal on Selected Areas in Communications*, 32(3):572–588, 2014.

- [49] Woongsup Lee, Lin Xiang, Robert Schober, and Vincent WS Wong. Direct electricity trading in smart grid: a coalitional game analysis. *IEEE Journal on Selected Areas in Communications*, 32(7):1398–1411, 2014.
- [50] Richard Southwell, Xu Chen, and Jianwei Huang. Quality of service games for spectrum sharing. *IEEE Journal on Selected Areas in Communications*, 32(3): 589–600, 2014.
- [51] Wei Zhong, Yixin Fang, Shi Jin, Kai-Kit Wong, Sheng Zhong, and Zuping Qian. Joint resource allocation for device-to-device communications underlaying up-link mimo cellular networks. *IEEE Journal on Selected Areas in Communications*, 33(1):41–54, 2015.
- [52] Mina Fahimi and Abdorasoul Ghasemi. Joint spectrum load balancing and hand-off management in cognitive radio networks: a non-cooperative game approach. *Wireless Networks*, pages 1–20, 2015.
- [53] Singapore LTA. Electronic road pricing (ERP). 2017. <https://www.lta.gov.sg/content/ltaweb/en/roads-and-motoring/managing-traffic-and-congestion/electronic-road-pricing-erp.html>.
- [54] AutoPASS. Find a toll station. 2017. <http://www.autopass.no/en/autopass>.
- [55] Mojang. Minecraft. 2017. <https://minecraft.net/en-us/>.
- [56] Noburu Kuno and Scarlet Schwiderski-Grosche. Presenting the winners of the Project Malmo Collaborative AI Challenge. 2017. <https://www.microsoft.com/en-us/research/blog/malmo-collaborative-ai-challenge-winners/>.
- [57] John Nicklow, Patrick Reed, Dragan Savic, Tibebe Dessalegne, Laura Harrell, Amy Chan-Hilton, Mohammad Karamouz, Barbara Minsker, Avi Ostfeld, Abhishek Singh, et al. State of the art for genetic algorithms and beyond in water

- resources planning and management. *Journal of Water Resources Planning and Management*, 136(4):412–432, 2009.
- [58] Peter W Sauer, Mangalore A Pai, and Joe H Chow. *Power System Dynamics and Stability: With Synchrophasor Measurement and Power System Toolbox*. John Wiley & Sons, 2017.
- [59] Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future generation computer systems*, 28(5):755–768, 2012.
- [60] Shankar Chakraborty. Applications of the moora method for decision making in manufacturing environment. *The International Journal of Advanced Manufacturing Technology*, 54(9):1155–1166, 2011.
- [61] Ronald L Graham and Pavol Hell. On the history of the minimum spanning tree problem. *Annals of the History of Computing*, 7(1):43–57, 1985.
- [62] Bernhard Korte, László Lovász, and Rainer Schrader. *Greedoids*, volume 4. Springer Science & Business Media, 2012.
- [63] LI Rongheng and Yue Minyi. A tighter bound for ffd algorithm. *Acta Mathematicae Applicatae Sinica*, 16(4):337–347, 2000.
- [64] Gerhard Reinelt. *The traveling salesman: computational solutions for TSP applications*. Springer-Verlag, 1994.
- [65] Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.
- [66] Stuart E Dreyfus. The art and theory of dynamic programming. Technical report, 1977.
- [67] Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Chapman & Hall/CRC, 2010.

- [68] Giorgio Ausiello, Pierluigi Crescenzi, Giorgio Gambosi, Viggo Kann, Alberto Marchetti-Spaccamela, and Marco Protasi. *Complexity and approximation: Combinatorial optimization problems and their approximability properties*. Springer Science & Business Media, 2012.
- [69] George B Dantzig. Discrete-variable extremum problems. *Operations research*, 5(2):266–288, 1957.
- [70] Kurt M Bretthauer and Bala Shetty. The nonlinear knapsack problem—algorithms and applications. *European Journal of Operational Research*, 138(3):459–472, 2002.
- [71] H Keller, U Pferschy, and D Pisinger. *Knapsack problems*. Springer-Verlag Berlin, 2004.
- [72] David Pisinger. A minimal algorithm for the bounded knapsack problem. *INFORMS Journal on Computing*, 12(1):75–82, 2000.
- [73] Ulrich Pferschy. Dynamic programming revisited: Improving knapsack algorithms. *Computing*, 63(4):419–430, 1999.
- [74] Silvano Martello and Paolo Toth. Branch-and-bound algorithms for the solution of the general unidimensional knapsack problem. *Advances in Operations Research*, 13(9):295–301, 1977.
- [75] Silvano Martello and Paolo Toth. *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc., 1990.
- [76] Marc Visée, Jacques Teghem, Marc Pirlot, and Ekunda L Ulungu. Two-phases method and branch and bound procedures to solve the bi-objective knapsack problem. *Journal of Global Optimization*, 12(2):139–155, 1998.
- [77] Matthew A Waller and Stanley E Fawcett. Data science, predictive analytics, and big data: a revolution that will transform supply chain design and management. *Journal of Business Logistics*, 34(2):77–84, 2013.
- [78] Julie A Marsh, John F Pane, and Laura S Hamilton. Making sense of data-driven decision making in education. 2006.

- [79] Weinan Zhang, Shuai Yuan, and Jun Wang. Optimal real-time bidding for display advertising. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 1077–1086, 2014.
- [80] Yada Zhu, Hongxia Yang, and Jingrui He. Co-clustering based dual prediction for cargo pricing optimization. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 1583–1592, 2015.
- [81] Jian Xu, Kuang-chih Lee, Wentong Li, Hang Qi, and Quan Lu. Smart pacing for effective online ad campaign optimization. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 2217–2226, 2015.
- [82] Wei Lu, Stratis Ioannidis, Smriti Bhagat, and Laks VS Lakshmanan. Optimal recommendations under attraction, aversion, and social influence. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 811–820, 2014.
- [83] Tamer Basar and Geert Jan Olsder. *Dynamic noncooperative game theory*, volume 23. Siam, 1999.
- [84] Centre for Co-operation with European Economies in Transition. *Glossary of industrial organisation economics and competition law*. Organization for Economic, 1993.
- [85] Robert J Aumann. The core of a cooperative game without side payments. *Transactions of the American Mathematical Society*, 98(3):539–552, 1961.
- [86] Lloyd S Shapley. A value for n-person games. *The Shapley value*, pages 31–40, 1988.
- [87] Ronald A Howard. Dynamic programming. *Management Science*, 12(5):317–348, 1966.
- [88] Lodewijk Cornelis Maria Kallenberg. Linear programming and finite markovian control problems. *MC Tracts*, 1983.

- [89] John L Nazareth and Ram B Kulkarni. Linear programming formulations of markov decision processes. *Operations research letters*, 5(1):13–16, 1986.
- [90] Doina Precup, Richard S Sutton, and Sanjoy Dasgupta. Off-policy temporal-difference learning with function approximation. In *ICML*, pages 417–424, 2001.
- [91] Hamid R Maei, Csaba Szepesvári, Shalabh Bhatnagar, and Richard S Sutton. Toward off-policy learning control with function approximation. In *ICML*, pages 719–726, 2010.
- [92] Barry D Nichols and Dimitris C Dracopoulos. Application of newton’s method to action selection in continuous state-and action-space reinforcement learning. *ESANN*, 2014.
- [93] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.
- [94] Richard S Sutton, David A McAllester, Satinder P Singh, Yishay Mansour, et al. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, volume 99, pages 1057–1063, 1999.
- [95] Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 21(4):682–697, 2008.
- [96] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *ICML*, pages 1889–1897, 2015.
- [97] Matthew Hausknecht and Peter Stone. Deep reinforcement learning in parameterized action space. In *ICLR*, 2016.
- [98] Peter Stone, Gal A Kaminka, Sarit Kraus, Jeffrey S Rosenschein, et al. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *AAAI*, pages 1504–1509, 2010.
- [99] Samuel Barrett, Peter Stone, Sarit Kraus, and Avi Rosenfeld. Teamwork with limited knowledge of teammates. In *AAAI*, pages 102–108, 2013.

- [100] Stefano V Albrecht, Jacob W Crandall, and Subramanian Ramamoorthy. Belief and truth in hypothesised behaviours. *Artificial Intelligence*, 235:63–94, 2016.
- [101] Finnegan Southey, Michael P Bowling, Bryce Larson, Carmelo Piccione, Neil Burch, Darse Billings, and Chris Rayner. Bayes’ bluff: Opponent modelling in poker. *arXiv preprint arXiv:1207.1411*, 2012.
- [102] Ronen I Brafman, Carmel Domshlak, Yagil Engel, and Moshe Tennenholtz. Planning games. In *IJCAI*, pages 73–78, 2009.
- [103] Jaume Jordán and Eva Onaindia. Game-theoretic approach for non-cooperative planning. In *AAAI*, pages 1357–1363, 2015.
- [104] Jaume Jordán, Mathijs de Weerd, Eva Onaindia, et al. A better-response strategy for self-interested planning agents. *Applied Intelligence*, pages 1–21, 2017.
- [105] Tuomas Sandholm. Abstraction for solving large incomplete-information games. In *AAAI*, pages 4127–4131, 2015.
- [106] Karel Horák, Branislav Bosanský, and Michal Pechoucek. Heuristic search value iteration for one-sided partially observable stochastic games. In *AAAI*, pages 558–564, 2017.
- [107] Sven Seuken and Shlomo Zilberstein. Formal models and algorithms for decentralized decision making under uncertainty. In *AAAMAS*, pages 190–250, 2008.
- [108] Noa Agmon and Peter Stone. Leading ad hoc agents in joint action settings with multiple teammates. In *AAMAS*, pages 341–348, 2012.
- [109] Mark Hansen. Airline competition in a hub-dominated environment: An application of noncooperative game theory. *Transportation Research Part B: Methodological*, 24(1):27–43, 1990.
- [110] Yoshinori Suzuki. The relationship between on-time performance and airline market share: A new approach. *Transportation Research Part E: Logistics and Transportation Review*, 36(2):139–154, 2000.

- [111] Wenbin Wei and Mark Hansen. Impact of aircraft size and seat availability on airlines demand and market share in duopoly markets. *Transportation Research Part E: Logistics and Transportation Review*, 41(4):315–327, 2005.
- [112] Gershon Alperovich and Yaffa Machnes. The role of wealth in the demand for international air travel. *Journal of Transport Economics and Policy*, 28(2):163–173, 1994.
- [113] Dipasis Bhadra and Jacqueline Kee. Structure and dynamics of the core us air travel markets: A basic empirical analysis of domestic passenger demand. *Journal of Air Transport Management*, 14(1):27–39, 2008.
- [114] Dipasis Bhadra and Michael Wells. Air travel by state its determinants and contributions in the United States. *Public Works Management & Policy*, 10(2):119–137, 2005.
- [115] Alberto Caprara, Hans Kellerer, Ulrich Pferschy, and David Pisinger. Approximation algorithms for knapsack problems with cardinality constraints. *European Journal of Operational Research*, 123(2):333–345, 2000.
- [116] BTS. Bureau of transportation statistics, 2016. URL http://www.rita.dot.gov/bts/data_and_statistics/by_mode/airline_and_airports/index.html.
- [117] BEA. Bureau of Economic Analysis. 2016. <http://www.bea.gov/regional/downloadzip.cfm>.
- [118] NTSB. National Transportation Safety Board. 2016. http://www.ntsbgov/_layouts/ntsb.aviation/index.aspx.
- [119] U.S. Census. Metropolitan and Micropolitan. 2016. <https://www.census.gov/population/metro/data/>.
- [120] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., 2006. ISBN 0387310738.

- [121] CE. Rasmussen and CKI. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, 2006.
- [122] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981. ISSN 0001-0782.
- [123] Peter J Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20: 53–65, 1987.
- [124] Robert L. Thorndike. Who belongs in the family. *Psychometrika*, 18(4):267–276, 1953.
- [125] David Arthur and Sergei Vassilvitskii. K-Means++: The advantages of careful seeding. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1027–1035, 2007.
- [126] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 226–231, 1996.
- [127] Eddy Caron, Frederic Desprez, and Adrian Muresan. Forecasting for grid and cloud computing on-demand resources based on pattern matching. In *CloudCom*, pages 456–463, 2010.
- [128] Amazon ec2 service level agreement. <https://aws.amazon.com/ec2/sla/>, 2016.
- [129] Microsoft Azure. Sla for virtual machines. https://azure.microsoft.com/en-us/support/legal/sla/virtual-machines/v1_2/, 2016.
- [130] Pankesh Patel, Ajith H Ranabahu, and Amit P Sheth. Service level agreement in cloud computing. 2009.
- [131] L. Kóczy. A recursive core for partition function form games. *Theory and Decision*, 63(1):41–51, 2007.

- [132] Lloyd S Shapley. A value for n-person games. Technical report, DTIC Document, 1952.
- [133] Georgios Chalkiadakis, Evangelos Markakis, and Nicholas R Jennings. Coalitional stability in structured environments. In *AAMAS*, pages 779–786, 2012.
- [134] Thanh H Nguyen, Amulya Yadav, Bo An, Milind Tambe, and Craig Boutilier. Regret-based optimization and preference elicitation for stackelberg security games with uncertainty. In *AAAI*, pages 756–762, 2014.
- [135] Chee Shin Yeo and Rajkumar Buyya. Service level agreement based allocation of cluster resources: Handling penalty to enhance utility. In *2005 IEEE International Conference on Cluster Computing*, pages 1–10. IEEE, 2005.
- [136] Omer F Rana, Martijn Warnier, Thomas B Quillinan, Frances Brazier, and Dana Cojocarasu. Managing violations in service level agreements. In *Grid Middleware and Services*, pages 349–358. Springer, 2008.
- [137] Linlin Wu, Saurabh Kumar Garg, and Rajkumar Buyya. Sla-based resource allocation for software as a service provider (saas) in cloud computing environments. In *CCGrid*, pages 195–204, 2011.
- [138] Shizuo Kakutani et al. *A generalization of Brouwer’s fixed point theorem*. Duke University Press, 1941.
- [139] Tuomas Sandholm, Kate Larson, Martin Andersson, Onn Shehory, and Fernando Tohmé. Coalition structure generation with worst case guarantees. *Artificial Intelligence*, 111(1):209–238, 1999.
- [140] Ning Chen, Pinyan Lu, and Hongyang Zhang. Computing the nucleolus of matching, cover and clique games. In *AAAI*, 2012.
- [141] Andreas S Schulz and Nelson A Uhan. Approximating the least core value and least core of cooperative games with supermodular costs. *Discrete Optimization*, 10(2):163–180, 2013.

- [142] Talal Rahwan, Tomasz Michalak, Michael Wooldridge, and Nicholas R. Jennings. Anytime coalition structure generation in multi-agent systems with positive or negative externalities. *Artificial Intelligence*, 186:95 – 122, 2012. ISSN 0004-3702. doi: <https://doi.org/10.1016/j.artint.2012.03.007>. URL <http://www.sciencedirect.com/science/article/pii/S0004370212000288>.
- [143] Amazon. Amazon ec2 pricing. <https://aws.amazon.com/ec2/pricing/>, 2016.
- [144] Investopedia. What is considered a healthy operating profit margin? <http://www.investopedia.com/ask/answers/041415/what-considered-healthy-operating-profit-margin.asp>, 2015.
- [145] Dusica Joksimovic, Michiel CJ Bliemer, Piet HL Bovy, and Zofia Verwater-Lukszo. Dynamic road pricing for optimizing network performance with heterogeneous users. In *ICNSC*, pages 407–412, 2005.
- [146] Chung-Cheng Lu, Hani S Mahmassani, and Xuesong Zhou. A bi-criterion dynamic user equilibrium traffic assignment model and solution algorithm for evaluating dynamic road pricing strategies. *Transportation Research Part C: Emerging Technologies*, 16(4):371–389, 2008.
- [147] Kuilin Zhang, Hani S Mahmassani, and Chung-Cheng Lu. Dynamic pricing, heterogeneous users and perception error: Probit-based bi-criterion dynamic stochastic user equilibrium assignment. *Transportation Research Part C: Emerging Technologies*, 27:189–204, 2013.
- [148] Guni Sharon, Josiah P Hanna, Tarun Rambha, Michael W Levin, Michael Albert, Stephen D Boyles, and Peter Stone. Real-time adaptive tolling scheme for optimized social welfare in traffic networks. In *AAMAS*, pages 828–836, 2017.
- [149] Takashi Akamatsu. Cyclic flows, markov process and stochastic traffic assignment. *Transportation Research Part B: Methodological*, 30(5):369–386, 1996.
- [150] Jean-Bernard Baillon and Roberto Cominetti. Markovian traffic equilibrium. *Mathematical Programming*, 111(1-2):33–56, 2008.

- [151] BPR. Traffic assignment manual. *US Department of Commerce*, 1964.
- [152] Hong K Lo and Wai Yuen Szeto. A methodology for sustainable traveler information services. *Transportation Research Part B: Methodological*, 36(2):113–130, 2002.
- [153] Hong K Lo, CW Yip, and KH Wan. Modeling transfer and non-linear fare structure in multi-modal network. *Transportation Research Part B: Methodological*, 37(2):149–170, 2003.
- [154] Hai-Jun Huang and Zhi-Chun Li. A multiclass, multicriteria logit-based traffic equilibrium assignment model under atis. *European Journal of Operational Research*, 176(3):1464–1477, 2007.
- [155] Richard S Sutton and Andrew G Barto. Reinforcement Learning: An Introduction, 2011.
- [156] Piotr Olszewski. Comparison of the hcm and singapore models of arterial capacity. In *TRB Highway Capacity Committee Summer Meeting*. Citeseer, 2000.
- [157] Government of Singapore. Department of statistics, Singapore. 2017. <http://www.singstat.gov.sg/>.
- [158] A place for mom inc. 2016 alzheimers statistics, 2017. URL <http://www.alzheimers.net/resources/alzheimers-statistics/>.
- [159] Jennifer Boger, Jesse Hoey, Pascal Poupart, Craig Boutilier, Geoff Fernie, and Alex Mihailidis. A planning system based on markov decision processes to guide people with dementia through activities of daily living. *IEEE Transactions on Information Technology in Biomedicine*, 10(2):323–333, 2006.
- [160] Jesse Hoey, Pascal Poupart, Axel von Bertoldi, Tammy Craig, Craig Boutilier, and Alex Mihailidis. Automated handwashing assistance for persons with dementia using video and a partially observable markov decision process. *Computer Vision and Image Understanding*, 114(5):503–519, 2010.

- [161] Jesse Hoey, Craig Boutilier, Pascal Poupart, Patrick Olivier, Andrew Monk, and Alex Mihailidis. People, sensors, decisions: Customizable and adaptive technologies for assistance in healthcare. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 2(4):20, 2012.
- [162] Samuel Barrett, Peter Stone, and Sarit Kraus. Empirical evaluation of ad hoc teamwork in the pursuit domain. In *AAMAS*, pages 567–574, 2011.
- [163] Zhaodong Wang and Matthew E Taylor. Improving reinforcement learning with confidence-based demonstrations. In *IJCAI*, pages 3027–3033, 2017.
- [164] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, pages 249–256, 2010.
- [165] Nikolaus Hansen. The cma evolution strategy: a comparing review. *Towards a new evolutionary computation*, pages 75–102, 2006.
- [166] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.
- [167] Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous deep q-learning with model-based acceleration. In *ICML*, pages 2829–2838, 2016.
- [168] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *ICML*, pages 1928–1937, 2016.
- [169] Oron Anschel, Nir Baram, and Nahum Shimkin. Averaged-dqn: Variance reduction and stabilization for deep reinforcement learning. In *ICML*, pages 176–185, 2017.
- [170] Terry Anderson. *The Theory and Practice of Online Learning*. Athabasca University Press, 2008.
- [171] Sam Ganzfried and Tuomas Sandholm. Endgame solving in large imperfect-information games. In *AAMAS*, pages 37–45, 2015.

-
- [172] Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10:1633–1685, 2009.
- [173] Haipeng Chen, Bo An, Dusit Niyato, Yeng Chai Soh, and Chuanyan Miao. Workload factoring and resource sharing via joint vertical and horizontal cloud federation networks. *IEEE Journal on Selected Areas in Communications*, 35(3): 557–570, 2017.
- [174] Haipeng Chen, Bo An, Guni Sharon, Josiah P Hanna, Peter Stone, Chunyan Miao, and Yeng Chai Soh. Dyetc: Dynamic electronic toll collection for traffic congestion alleviation. In *AAAI*, 2018.
- [175] Yanhai Xiong, Haipeng Chen, Mengchen Zhao, and Bo An. Hogrider: Champion agent of microsoft malmo collaborative ai challenge. In *AAAI*, 2018.