Game-theoretic Payoff Allocation in Multiagent Machine Learning Systems

Dongge Han
Balliol College
University of Oxford

Thesis for
Doctor of Philosophy

Michaelmas Term 2021

To my parents,
Ronglan and Jie

Acknowledgements

First and foremost, I would like to thank my supervisors, Prof. Michael Wooldridge and Prof. Alex Rogers, for guiding me to appreciate and explore the fascinating world of multiagent systems and game theory, and for their invaluable advice and continuous support for my research throughout my DPhil study. Many thanks to Prof. Edith Elkind for her guidance on cooperative game theory at the beginning of my DPhil.

I am thankful to my viva examiners Ani Calinescu and Georgios Chalkiadakis for their insightful and thought-provoking discussions and for their thoughtful comments that really help me to improve and polish the thesis. Many thanks to my transfer and confirmation examiners Ani Calinescu, Shimon Whiteson, and Thomas Lukasiewicz for their kind guide and help in shaping my research directions.

I am grateful to my collaborators who I am fortunate to work with. I am deeply thankful to Sebastian Tschiatschek for his invaluable help and guidance, and for generously sharing his knowledge and advice on the mathematically beautiful theory of submodularity, and the interesting connection between inverse and hierarchical RL. I am very thankful to Wendelin Böhmer for his helpful discussions on hierarchical and multiagent RL and his guidance on RL problem formulations. I am so thankful to Tomasz Michalak and Chris Lu for the fascinating discussions on applying cooperative game theory to continuous control in robotics. Finally, I am really grateful to Paul Harrenstein for his insightful and interesting discussions on finding equilibrium solutions in weighted Boolean games.

Warm thanks go to my friends and colleagues in the Wolfson Building, Ayumi Igarashi, Xin Zhou, James Paulin, Jiarui Gan and Paul Harrenstein. I am grateful to Sebastian Tschiatschek, Olya Ohrimenko, Ryota Tomioka, Tom Voice, David Vandyke, Tom Gunter and Thor Helgason, and my fellow interns for their helpful advice during my internships at Microsoft Research and Apple Siri in Cambridge. I would like to thank the staff at Balliol College and administrators in the CS Department, Julie Sheppard, Sarah Retz-Jones, Lyn Hambridge, Jayne Bullock and Jenny Dollard for their ongoing help and support. A warm thank to my friend Yulan for always giving me her support and advice. I am grateful to my boyfriend Xuan for his love, support and guidance throughout the DPhil.

Last but not least, to my parents, Ronglan and Jie who have always believed in me, inspired me and encouraged me. I am fortunate to have such wonderful parents. Thank you for your unconditional love!

Abstract

Machine learning (ML) is becoming ubiquitous in real-world applications, spanning from our domestic devices such as vacuum cleaner robots, smart phones, virtual assistants, to industrial applications such as manipulators, warehouse robots, to public services such as medical imaging, surveillance cameras, energy grid, etc. With the growing ubiquity and interconnection of these devices, interactions among them will soon become commonplace. These emerging ML problems and the interactions among agents can be formulated as Multiagent ML Systems. On the one hand, many complex ML tasks require the cooperation among multiple agents carrying distributed resources and capabilities. On the other hand, multiagent solutions often lead to improved efficiency, robustness and scalability. Among the many aspects of multiagent ML systems, an important research problem is payoff allocation. This is because multiagent ML systems typically receive a global payoff for the overall performance of all agents, while a fine-grained evaluation of each agent's contribution is absent. Nevertheless, these agent-specific payoffs not only offer natural incentives for their contribution and cooperation, but also provide crucial feedback signals for interpreting the agent's contributions and improving the their future cooperative policies. In this thesis, we investigate the properties of emerging multiagent ML systems and address new challenges that arise when applying classic payoff allocation methods from cooperative game theory to them.

The first part of the thesis investigates payoff allocation in submodular multiagent ML problems. Though convex games are commonly studied in cooperative game theory, many multiagent ML applications naturally exhibit submodular characteristics. Moreover, properties of the ML applications give rise to increased computational burden, as well as a new type of malicious attack (i.e., the replication manipulation). To address these challenges, we present a theoretical analysis on payoff allocation methods (in particular, semivalues) in submodular cooperative games, and characterise their robustness against the replication manipulation. We then extend our theoretical results to an emerging application of ML data markets and discuss related and more complex attack models. Moreover, we present a sampling method for estimating the payoff allocation methods to address the computational issue. Finally, we empirically validate our results on a classic facility location problem and on real-life machine learning datasets.

In the second part of the thesis, we investigate the integration of the payoff allocation into the learning process for guiding multiagent reinforcement learning (MARL) updates. To this end, we present a framework of game-theoretic payoff allocation in MARL for robotic control. Though the payoffs to agents provide crucial reward signals for improving their cooperative policy, a new challenge arises when performing the payoff allocations during learning. That is,

the characteristic values of coalitions are absent except for the grand coalition, i.e., the global reward. On the one hand, it is impossible to re-run the simulation for all possible coalitions at each step. On the other hand, brute-force inferring the value of the unseen coalitions would suffer from highly inaccurate estimations due to high dimensional continuous state and action spaces in the robotic application. To address this challenge, we further incorporate a model-based RL module into our payoff allocation framework, and use a learned model of the environment to simulate values of the unseen coalitions. We empirically demonstrate that our model-based payoff allocation framework greatly improves the performance and sample-efficiency of the MARL system on (multiagent) MuJoCo robotic locomotion control tasks.

In conclusion, the results presented in this thesis pave the way towards the application of game-theoretic payoff allocation methods in emerging multiagent ML systems, and opens up a number of exciting directions for future research in this topic.

Contents

Appendices

List of Figures

List of Tables

List of Abbreviations

CTDE . . . . . Centralised Training, Decentralised Execution.

DDPG . . . . . Deep Deterministic Policy Gradient.

FLP . . . . . Facility Location Problem.

GAE . . . . . Generalised Advantage Estimation.

IQL . . . . . Independent Q Learning.

LOO . . . . . Leave-one-out.

MAPPO . . . . . Multiagent Proximal Policy Gradient.

MARL . . . . . Multiagent Reinforcement Learning.

MAS . . . . . Multiagent System.

MB, MBRL . . . Model-based, Model-based Reinforcement Learning.

MDP . . . . . Markov Decision Process.

ML . . . . . Machine Learning.

MLP . . . . . Multi-layer Perceptron.

PPO . . . . . Proximal Policy Gradient.

RL . . . . . Reinforcement Learning.

TD . . . . . Temporal Difference.

TRPO . . . . . Trust Region Policy Gradient.

List of Mathematical Notations

Game Theory Notations (Chapter 3)

i, c, k	Indices of players, coalition size, number of replications
N	A set of all players
C	Coalition of players,
C^R, N^R	Replica players, all players after replication
$v(C)$	Characteristic value of coalition C
$G = (N, v)$	Cooperative game with players N and characteristic function v.
$\mathcal{MC}_i(C)$	Marginal contribution of agent i towards coalition C
$\mathcal{L}$	A set of facility locations
D	A set of customers, or a set of data
u_{id}	Utility function of facility location i to customer d
φ_i	Payoff allocated to player i
φ_i^{tot}	Total payoff of the replicating player
$\varphi_i^{\text{Shapley}}, \varphi_i^{\text{Banzhaf}}, \varphi_i^{\text{LOO}}$	The Shapley value, Banzhaf value and Leave-one-out value of player i
$w_{c,N}$	Weight of the solution concept for coalitions of size c
α_c	Importance weight towards coalition size c
α_c^k	Importance weight towards coalition size c after k replications
$z_i(c)$	Average marginal contribution of player i towards size c coalitions

Reinforcement Learning Notations (Chapter 4)

i, t	Indices for agents, timesteps
s_t, a_t	State and action at time t
π^i	Policy of agent i (Sec. 2.2, 4.3.3)
G_t	Discounted future return from t onwards (Sec. 2.2)
$V^\pi(s), Q^\pi(s, \mathrm{a})$	State and action-value functions (Sec. 2.2.1.2)
$A^\pi(s, \mathrm{a})$	Advantage function (Sec. 2.2.3.1, 4.3.3)
ω, θ, ϕ_s, ϕ_r	model parameters of value function, policy, and model V, π, f_s, f_r (Sec. 4.3.3, 4.5)

$v^C(s, \mathrm{a})$ . . . . . Characteristic value of coalition C (Sec. 4.4.1)

$\tilde{\mathrm{a}}_t$ . . . . . Actions a_t masked by coalitions (Sec. 4.4.1)

$\mathcal{MC}^i(C, s_t, \mathrm{a}_t)$ . . . Marginal contribution of agent i to C at state s_t and taking action a_t.

$\varphi^i(v)$, α_c . . . . . Semivalues and the importance weights (Sec. 4.4.2)

$f_s(s, \mathrm{a}), f_r(s, \mathrm{a})$ . . . Dynamics and reward models (Sec. 4.5.1)

$\hat{s}_{t+1}, \hat{r}_t$ . . . . . States/Rewards predicted by model (Sec. 4.5.1)

Please note that in Chapter 3, agent indices i appear in the subscripts following the convention of cooperative game theory. In Chapter 4, timesteps t are included as subscripts, and agent indices i appear as superscripts following the convention in multiagent reinforcement learning.

1 Introduction

Contents

1.1 Multiagent Machine Learning Systems

As a quest towards artificial intelligence (AI), machine learning (ML) is the research field that enables machines to learn automatically from data in order to perform predictions and optimal decision making. Empowered by the availability of large quantities of data and computational resources, machine learning is becoming ubiquitous in real-world applications, spanning from our domestic devices such as vacuum cleaner robots, smartphones, virtual assistants, to industrial applications such as manipulators, warehouse robots, to public services such as medical imaging, surveillance cameras, energy grid, etc. With the growing ubiquity and interconnection of ML devices, interactions among these entities will soon become commonplaces.

1.1.1 Multiagent Systems of Machine Learning Agents

Take autonomous driving as an example, an autonomous vehicle nowadays can learn to perceive the world, plan its trajectory and perform optimal control by leveraging cutting-edge machine learning techniques such as reinforcement learning (RL). In the foreseeable future, many such vehicles will operate on the road at the same time, hence their interaction, communication, and coordination would play essential roles for the safety and efficiency for the future transport system [129, 87].

A *multiagent system* (MAS) [148] is often used to model the interactions among multiple autonomous agents and their environment. Specifically, an agent is an abstract model which can represent a large category of entities. For example, an agent can represent a physical entity such as a seller in a data market, a device in a federated learning system, a robot in a rescue team, an autonomous vehicle; or a digital entity such as a software agent that plays chess, a node in an online social network, or a virtual assistant. In the machine learning community and in particular the subfield of interpretable ML, the notion of agents is also extended to represent passive entities that collaboratively contribute towards a task, such as data or features that contribute to the training of a machine learning model. According to the represented entities and their decision-making approaches, agents can be designed to have shared or different attributes such as sensor observations, actions, capabilities and policies. Moreover, depending on the specific problem setting, agents in a multiagent system may hold different or shared goals and preferences. A common approach to modelling the preferences of an agent is to define a *utility function*, which is a quantitative evaluation of different states in the environment, and the agent will aim to reach the states with higher utility. As agents may represent varying types of physical or digital entities, we will adopt the pronouns *it* and *they* when referring to a single or group of agents.

Having defined the individual agents, the next step is to determine the mode of their interactions. To this end, we may ask questions such as whether the agents can communicate with each other through a communication graph? Do the agents have the ability to learn and adapt from their interactions. Are the agents benevolent or competitive [129]?

On the one hand, the agents can be selfish (rational) and only aim to achieve their own goals. This assumption of rationality is common in AI and economics, and is the foundation of many important game-theoretic solution concepts such as the Nash Equilibrium [95]. Interestingly, the assumption of rational agents can even hold in cooperative scenarios, where an agent only

cooperates with others in order to achieve its own goals. For example, consider a multi-party machine learning problem where a group of agents each holds a different piece of resource. Here the agents can be a group of hospitals each with limited medical data, a group of data providers in a data market, or a mobile device in a federated learning network. While the agents collaboratively pool their resources in order to train a joint machine learning model, each agent can be considered rational, that is, it is only interested in maximising its own utility, be it the performance of the trained ML model for its own downstream task, or the monetary compensation for its contributed resource. In the extreme case of competitive interaction, the agents can fall into a zero-sum situation where optimising their own utilities correspond to oppressing the others'. An example is the game of Go, where the win of an agent means the loss of the opponent.

On the other hand, agents may be benevolent and willing to help each other to achieve their different goals. In the extreme case, the agents can be fully-cooperative and all agents aim to achieve a shared objective. In fact, many complex real-world applications that require cooperation among a team of agents can be modelled by a fully-cooperative multiagent system. An example is the robot rescue teams, where the robots cooperate to assist in the search and rescue of victims following disastrous situations. Another common example is multiagent manipulation, where two or more robotic agents (e.g., robotic arms or quadruped robots) collaboratively grasp and move an object towards a target location and rotate the object towards a specified configuration. On a lower granularity, each robot with multiple components can also be modelled as a fully-cooperative multiagent system. Here the agents can correspond to joints or actuators, which jointly control the whole robot to perform locomotion.

1.1.2 Payoff Allocation for Multiagent Machine Learning Systems

In real-world multiagent ML systems and especially cooperative scenarios, the agents often receive a global payoff for their joint performance, while a detailed attribution of the payoff to each agent is absent. This makes it difficult to provide agent-specific feedback. For example, some agents may have contributed significantly towards the task, while some agents may have under-performed, being the bottlenecks of the system. Imagine a traffic system of autonomous driving vehicles. In the case of traffic congestion, it is useful to obtain the agent-specific contributions and identify

which of the vehicles were the major bottlenecks, then use this feedback to adjust their behaviors and improve the agents' contributions towards the system.

To this end, an important aspect and great challenge of multiagent machine learning is payoff allocation [84, 85, 2, 42, 75, 108], which evaluates the contribution from each member towards the multiagent system and how much payoff they are entitled to. On the one hand, payoffs (e.g., monetary compensations) to agents according to their contributions provide incentives for the agents to increase their contributions towards the cooperative task. On the other hand, the payoffs allocated to agents can be used as crucial feedback signals for them to learn and improve their cooperative behaviours. In the following, we describe payoff allocation in two types of real-world applications.

1. Payoff Allocation for Data Valuation. Training machine learning models towards real-world applications may require large volumes of high quality, up-to-date and application-specific training data. However, manually collecting such training data can be time-consuming and error-prone for ML practitioners. In order to acquire the amount of data that is sufficient for training the ML model, they may need to purchase data from one or multiple data providers. Alternatively, multiple agents can collaborate to train a machine learning model, either in a centralised approach (e.g., by pooling their data) or in a decentralised approach (e.g., federated learning). Often a payoff such as monetary compensation is allocated to each data provider according to their contribution, which rewards their participation and incentivizes the provision of high-quality data. Though data nowadays is becoming an important digital asset, there are no simple or universal methods for data valuation, especially in the emerging field of multiagent machine learning which is gaining increasing prominence in real-world applications and wide research interests. This calls for a payoff allocation mechanism for data valuation, which evaluates the contribution of the data from each data provider towards the trained ML model.

2. Payoff Allocation as a Reward Signal. Imagine a robot manipulation task where three robotic agents A, B, C cooperate to move an object. Collectively, the agents successfully delivered the object to the target location and received a reward. However, if we closely examine the execution process, only agent A consistently pushed the object towards the correct direction. Meanwhile, agent B constantly pushed the object in the wrong direction, while agent C moved randomly and did not assist in moving the object. In this case, how can we decide how much

reward shall be allocated to each agent? Should Agent A receive the whole reward while B and C get punished? This example illustrates a payoff allocation problem in multiagent reinforcement learning (MARL). In reinforcement learning, an agent learns to perform a task through repeated interactions with the environment. To learn the correct behaviour, the agent uses the reward feedback on its behaviours in the previous interactions to improve its policies. Therefore, the reward signals are crucial for the agents' learning as they implicitly characterise the task objective for the agents. In a cooperative multiagent system, however, oftentimes only a global reward signal is available which evaluate the performance of the joint actions of all agents, whereas the agent-specific reward is absent. In the above example, if agents B and C receive the global reward for their task completion, the agents will update their model in the direction which reinforces their false behaviours, hence the lack of credit assignment may lead to agents learning in the wrong directions.

The above examples demonstrate some of the important roles played by payoff allocation in multiagent machine learning systems. Apart from these applications, payoff allocation is also useful in many other machine learning applications such as feature importance analysis [84], which provides quantitative measurements for the contributions of each feature towards a machine learning model. More details on the related applications will be discussed in the future work chapter.

1.1.3 Cooperative Game Theory for Multiagent Machine Learning Systems

As we have seen above the important applications of payoff allocation, a natural question then arises: across these different emerging applications, are there generic methods for evaluating the contribution of the agents, and what are the criteria for assessing the payoff allocation methods? To answer this question, we look at cooperative game theory, where payoff allocation has been extensively studied. For example, the most popular and widely adopted solution concept for payoff allocation is the Shapley value [120]. Introduced by Shapley in his 1953 paper, the Shapley value defines a way to allocate the joint payoff of the multiagent system to each member. Along with the allocation method, the Shapley value comes with some natural properties which are commonly used as the criteria of fairness for general payoff allocation methods. In the rest of the thesis, we will extend from the Shapley value and study a class of Shapley-like solution concepts called the Semivalues [32]. A major novel contribution of this thesis is *applying solutions from classic cooperative game theory to the multiagent machine learning systems*. On the one hand,

cooperative game theory provides a principled approach to payoff allocation in multiagent systems. On the other hand, the use of machine learning techniques enables the multiagent systems to be applied to real world applications. However, applying cooperative game theory to multiagent machine learning is non-trivial. For example, in classic cooperative games, supermodularity is often assumed as a natural incentive for the cooperation among (rational) agents. By contrast, in many real-world ML applications, submodularity commonly happens (e.g., redundancy caused by overlapping information) and the cooperation among agents can still form due to the cooperative nature of the applications (e.g., multi-robot systems). As a result, research on the theoretical aspects of game theoretic solutions in submodular ML systems is urgently needed. In this thesis, we aim to address some of these challenges that may arise when applying classic cooperative game theory to the multiagent machine learning systems. In the following section, we present an overview of the research questions and an outline of the rest of the thesis.

1.2 Research Questions and Thesis Outline

In this thesis, we study some of the interesting properties and address new challenges that arise when *applying classic game-theoretic payoff allocation methods to the multiagent machine learning systems*. In particular, we focus on payoff allocation among agents in cooperative scenarios, and we look into the two aforementioned applications, 1. payoff allocation for data valuation in ML data marketplaces; 2. payoff allocation as a reward signal for multiagent RL. What it follows outlines topics studied in the rest of the thesis, which are also shown in the roadmap in Figure 1.1.

1.2.1 Payoff Allocation and Replication Manipulations in Submodular Games

A large body of cooperative game theory research are devoted to convex games [121], where agents are more useful when joining a larger group of other agents. However, many multiagent machine learning systems exhibit submodular or approximate submodular behaviours. This is due to the natural properties of the interaction among agents. An example is the collaborative machine learning system where agents pool their training data to jointly train an ML model. The data held by each agent often carry redundant or partially overlapping information towards the ML task. As a result, the performance of the trained model would be approximately submodular with

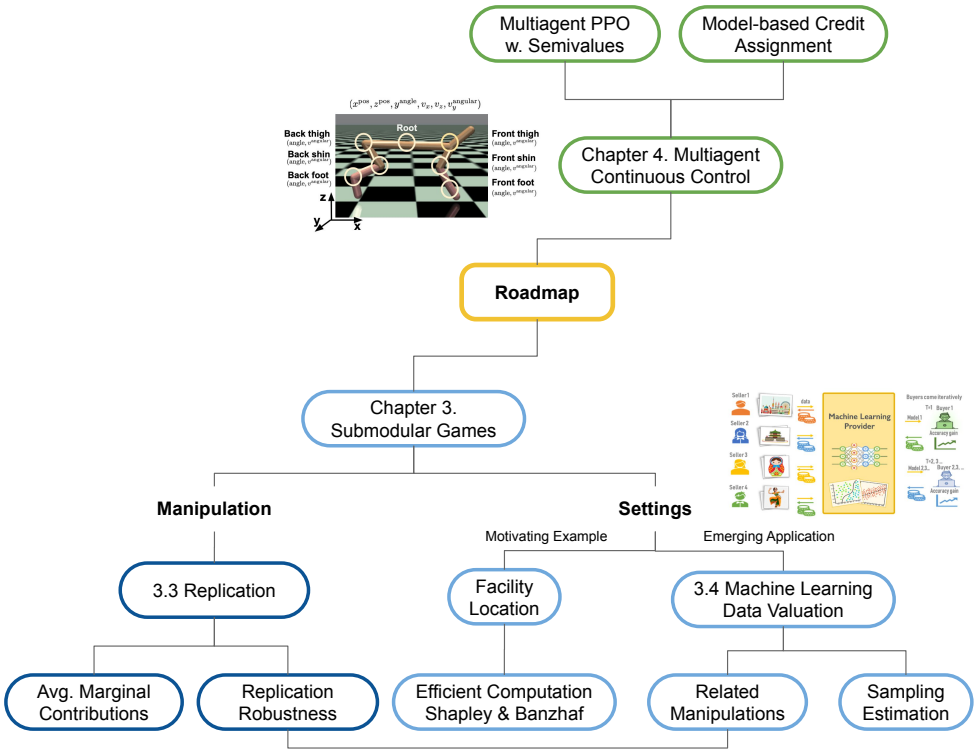


Figure 1.1: Thesis Roadmap

respect to the data, i.e., with the growing amount of data, adding more data will not significantly improve the model performance. Depending on the specific objective function, the submodularity property can also apply to other multiagent ML systems such as a set of ML features [29] or a team of robots [154]. Motivated by the wide range of submodular problems in multiagent ML applications, we will study some of the characteristics of game-theoretic payoff allocation methods in submodular cooperative games.

Apart from submodularity, we observed that the properties of machine learning systems give rise to new ways of malicious manipulations. In cooperative games, malicious manipulations have previously been studied in the context of voting games, where agents aim to increase their voting power by performing malicious splitting/merging manipulations [4]. In the emerging multiagent machine learning systems, we observe a new type of malicious manipulation, i.e., replication manipulation. This is because many digital resources such as ML data and ML features can be replicated, hence agents can easily replicate their resources and act under multiple false identities. We are therefore interested to ask: What governs the behaviours and robustness of

payoff allocation methods against the new replication manipulation in submodular cooperative games? And are there mechanisms to counteract the replication manipulation?

Furthermore, a practical issue around payoff allocation in multiagent machine learning systems is the expensive computation. For example, to compute the Shapley value for the agents, we need to compute the marginal contribution of each agent towards all possible coalitions of other agents. Imagine if the valuation of each coalition corresponds to training a machine learning model, then in total $2^{|N|}$ models need to be trained in order to compute the payoff allocation.

We address these questions in Chapter 3. Specifically, Section 3.3 studies the theoretical aspects of payoff allocation methods (in particular, semivalues) in submodular cooperative games and their robustness against the replication manipulation[1]. To list a few of our results, we derived closed-form solutions for the Shapley value and the Banzhaf value in a classic submodular facility location problem. And in general submodular games, we show that the average marginal contribution of an agent towards coalitions of other agents decreases with growing coalition size. In the context of replication, we present a necessary and sufficient condition that characterises the robustness of all semivalues against replication, e.g., the Shapley value is vulnerable to the replication manipulation, where an agent can increase its total payoff by replicating. In Section 3.4, we extend our theoretical study to an emerging application, namely, multiagent ML data markets, which connect data providers with ML practitioners who are in need of application-specific and up-to-date data. By modelling the market as a cooperative game, the agents (i.e., data providers) receive a payoff according to the contribution of their data, and a malicious agent may perform data replication to increase its payoff. Extending from our basic replication manipulation results, we also discuss more realistic manipulation scenarios such as a combination of data replication and perturbation, data splitting, or replication only on a subset of the data. Moreover, to address the expensive computation in the data market payoff allocation, we introduce an efficient sampling algorithm for estimating the semivalues, which demonstrates strong empirical improvements against state-of-the-art methods.

[1] For a detailed outline of our results please refer to Section 3.1.

1.2.2 Payoff Allocation in Robotic Multiagent Systems

In the previous research question, payoff allocation was applied to perform valuations of agents' contributions in multiagent machine learning systems. The payoff allocation is performed *post-hoc* and does not directly assist in the machine learning model updates. Meanwhile in a multiagent reinforcement learning system, the agent-specific payoffs are natural learning signals for guiding the agents' policy updates. In Chapter 4, we study payoff allocation as a reward signal for multiagent robotic control, and in particular, we study the continuous control of a single robot, where we model the robot by a multiagent system of its connected body components.

Reinforcement learning (RL) has shown great promise in robotic continuous control tasks. Nevertheless, prior research in this vein centers around the centralized learning setting that largely relies on the communication availability among all the components of a robot. However, agents in the real world often operate in a decentralised fashion without communication due to latency requirements, limited power budgets, and safety concerns. By formulating robot components as a system of decentralised agents, we present a decentralised multiagent RL framework for continuous control. To start with, we build a Cooperative Multiagent Proximal Policy Optimisation (PPO) framework that allows for centralized optimisation of the agents' policies during training and decentralised operation during execution. However, as discussed in the previous section (c.f., Section 1.1.2), the cooperative system only receives a global reward signal which is not attributed towards each agent. To address this challenge, we present a credit assignment RL framework based on Semivalues, which allocates to each agent a reward based on its contribution at each training step. The term *credit assignment* [43] is often used in RL while *payoff allocation* is commonly used in game theory [22]. In the rest of the thesis, we will use the terms credit assignment and payoff allocation interchangeably, as a real value assigned to an agent that quantifies its contribution in the team.

Though the game-theoretic framework allows us to effectively evaluate the agent-specific reward signal, a new challenge arises when applying the semivalues during the learning process as opposed to performing post-hoc analyses. That is, *the characteristic values of coalitions are absent except for the grand coalition (i.e., the global reward of the whole system).* So how can we obtain the value of different coalitions of agents? On the one hand, it is impossible to re-run the simulation for each possible coalition at each step, as the credit assignment needs to be performed at each

training step of training, which is typically on the scale of millions. On the other hand, brute-force inferring the value of the unseen coalitions would suffer from highly inaccurate estimations, as a result of high dimensional continuous state and action spaces. To address this challenge, we incorporate a model-based RL module into our credit assignment framework. Typically, model-based RL allows the agent to learn a model of the environment transitions in addition to its policy model. The environment model is then used to generate simulated experiences for training the agent's policy model. In this thesis, we integrate model-based RL to our multiagent system and use the environment model to simulate values of the unseen coalitions. We demonstrate that our model-based credit assignment framework greatly improves the performance and sample efficiency of the multiagent RL system on MuJoCo [140] robotic locomotion control tasks.

1.2.3 Other Projects Completed Towards the DPhil

Apart from payoff allocation, I also studied the following topics in RL related to the thesis.

1. Abstraction and Skill Transfer via Inverse Reinforcement Learning (IRL): IRL [1, 97, 135, 155] is an approach to learning from demonstrations. In this project, we applied IRL to perform transfer of skills (i.e., options [134]). Using the skill transfer framework, we further combine state and temporal abstractions which generates abstract SMDP's that can be used for efficient planning. In future work, it is interesting to apply IRL for multiagent RL, allowing agents to learn from each other.

2. Multiagent Coordination through Hierarchical RL: In a multiagent system, an agent's optimal policy depends on the policies of other agents. The options framework allows agents to take into account others' behaviours by broadcasting their current options. We further propose the dynamic termination of options to addresses the flexibility-predictability dilemma. In future work, it is interesting to further explore the opportunity to apply hierarchical RL for multiagent robotic continuous control.

3. Behavioural Equilibria in weighted Boolean games. In this project, we studied the computation of equilibria solutions in weighted Boolean games, where an agent aims to satisfy a weighted set of propositional logic goals by assigning truth values to its variables. However, an agent's goal may contain others' variables. We studied the strategic interactions of agents in such games, developing efficient algorithms for finding equilibrium solutions. In future work,

it is interesting to apply the Boolean game formulation for modelling heterogeneous multiagent RL systems where agents exhibit different objectives.

For more details of these projects please refer to publications listed in the following section.

1.2.4 A List of Papers Completed Towards the DPhil

The main contributions of the thesis are based on the following papers where I am the first author and the main contributor:

1. [54] **Han, D.**, Wooldridge, M., Rogers, A., Tople, S., Ohrimenko, O., and Tschiatschek, S. Replication-Robust Payoff-Allocation for Machine Learning Data Markets. (Currently under review for IEEE Transactions on Artificial Intelligence)

   This paper studies game-theoretic solution concepts and their robustness under a replication manipulation in submodular multiagent ML systems. We applied the theoretical results to data valuation in an ML data marketplace. The work in this paper is presented in Chapter 3 of the thesis.

2. [56] **Han, D.**, Lu, C., Tomasz, M., Wooldridge M., Multiagent Model-based Credit Assignment for Continuous Control. (Accepted as a full paper to AAMAS-22)

   This paper proposed a model-based game-theoretic credit assignment framework for continuous control in robot learning. The work in this paper is presented in Chapter 4 of the thesis.

The following are a list of my first-authored papers which are completed during the course of the DPhil and are relevant to the topics covered in the thesis. These papers are omitted for the coherence of the thesis:

1. [53] **Han, D.**, Boehmer, W., Wooldridge, M., and Rogers, A. Multi-agent Hierarchical Reinforcement Learning with Dynamic Termination. (Published in AAMAS-19 as an extended abstract. Full paper published in Pacific Rim International Conference on Artificial Intelligence, 2019).

   This paper studied coordination in multiagent RL. We proposed a multiagent hierarchical RL framework which allows agents to communicate their current subgoals, moreover, a

termination operator which learns to dynamically balance the agents' predictability and flexibility for improved coordination.

2. [57] **Han, D.**, Wooldridge, M., and Tschiatschek, S. MDP Abstraction with Successor Features. (Accepted in AAAI-22 Workshop on Reinforcement Learning in Games).
   This paper proposed an abstraction framework for Semi-Markov Decision Processes in hierarchical RL. The framework unifies state and action abstraction, and allows agents to transfer and efficiently plan with their learned skills in new environments through inverse RL.

3. [55] **Han, D.**, Harrenstein, P., Nugent, S., Philpott, J., and Wooldridge, M. Behavioural Strategies in Weighted Boolean Games. (Published in Information and Computation, 2021).
   This paper studies a type of non-cooperative game called Weighted Boolean Game. Agents strategically assign values to Boolean variables in order to achieve their own objectives expressed as propositional logic formulas. We proposed and computed the equilibrium solutions for the Boolean variable assignments.

2
Background

Contents

In this chapter, we will introduce the background knowledge necessary for understanding the rest of the thesis. We will first lay the theoretical foundation of this thesis by introducing the concepts and notations of cooperative game theory in Section 2.1. Specifically, we will briefly introduce multiagent problem solving in Section 2.1.1, we then introduce cooperative games and common payoff allocation solution concepts with illustrative examples in Section 2.1.2. In Section 2.1.3, we take the perspective of the players (agents) and discuss common malicious manipulations taken by a single or group of players in order to optimise their objective. We then introduce the concepts of reinforcement learning (RL) in Section 2.2, including the basic concepts and notations in Section 2.2.1, followed by model-free RL topics in Section 2.2.2. We then introduce continuous control RL methods, including Proximal Policy Gradient (PPO), upon

which we will develop our multiagent RL framework. Finally, we introduce the formulations of multiagent RL in Section 2.2.4 and model-based RL in Section 2.2.5.

2.1 Multiagent Systems and Cooperative Game Theory

A *Multiagent System (MAS)* is a system composed of multiple interacting intelligent agents [148], which can often be used to solve problems that are difficult or impossible for an individual agent. Multiagent problem solving often includes topics such as cooperation, coordination and communication, negotiation.

2.1.1 Multiagent Problem Solving and Games

There is a substantial body of classic literature on the formulation and optimisation of multiagent problems. Here we give a brief overview of some of the existing frameworks which cover different types of agent interactions.

For cooperative multiagent systems, a classic body of literature is Cooperative Distributed Problem Solving (CDPS). As a subfield of Distributed AI, CDPS focus on how a loosely-coupled network of agents can work together and cooperate in order to achieve a common goal that is often beyond their individual capabilities [33, 148]. CDPS embodies a wide range of multiagent coordination approaches such as negotiation, cooperation under inconsistencies, organisational structuring, and multiagent planning. An example is the Distributed Constraint Optimisation Problem (DCOP) [39], which is defined by a group of agents, a set of variables each belongs to an agent, and constraints on the values assigned to the variables. And the aim of the agents is to find variable assignments that minimise the cost of the constraints in a decentralised manner.

Multiagent planning extends multiagent problem solving to sequential settings. Markov Decision Processes (MDP) is the most common model for single-agent decision making under uncertainty. MDP's can be extended to model multiagent decision-making problems, such as the frameworks of multiagent MDP's (MMDP) [12] for fully-observable settings, and Decentralised Partially Observable MDP's (DEC-POMDP) [10] for partially-observable scenarios. Furthermore, in cases where a model of the environment is not available for planning, we can facilitate agents with learning capabilities. Specifically, multiagent reinforcement learning (MARL) [12] extends

multiagent planning problems to agents that can learn to perform decision making through repeated interactions with the environment.

When agents are not fully-cooperative and each holding different goals, they need to act strategically in order to maximise their own utilities. *Game Theory*, a branch of mathematics studying the mathematical models of strategic interactions among rational (self-interested) decision-makers [22], provides a good theoretical formulation for studying strategic multiagent decision making. In particular, a *game* is a mathematical model to capture the environment in which the decision-makers interact. These decision makers are termed as *agents*, or *players* of the game. [1]

There are two important types of games studied in game theory – *non-cooperative games* and *cooperative games*. The class of non-cooperative games focus on individual agent's strategies whereas the cooperative games focus on the formation of coalitions and payoff allocations. For non-cooperative games, simultaneous gameplay is often represented by a normal form game, that is, a matrix that represents all possible combinations of the agents' strategies and the utilities of each agent corresponding to the outcomes of each combination. A classic example is the Prisoner's Dilemma [107], where two prisoners separately choose whether or not to betray each other, and each player receives a sentence that is based on the outcome of their collective choices. Many solutions are developed for non-cooperative games, the most common solution concept being the Nash equilibrium [95], which corresponds to equilibrium points where all players are playing their best response hence no agent has the incentive to deviate from its current strategy. For sequential scenarios, the gameplay can be represented by an extensive form game, i.e., a game tree. To solve the extensive form games, there are equilibrium and sequential equilibrium solutions which can be computed using linear programming [123]. A classic sequential game is the game of Go, and in 2015, the algorithm AlphaGo [125] achieved super-human performance using a combination of reinforcement learning and Monte-Carlo Tree Search.

In contrast to non-cooperative games where agents make individual decisions, cooperative games are concerned with the collective behaviours of agents and their payoff allocations. For example, in cooperative games, agents can form coalitions with others, and decide how much payoff each member in the coalition is entitled to, while the detailed gameplay is omitted and the utility of different coalitions are often provided as an oracle function. In the rest of the thesis,

[1] We will use these two terms *Agents* and *Players* interchangeably in the rest of the thesis. We will mainly use *players* in the context of cooperative games, and *agents* in the context of multiagent reinforcement learning.

we are interested in applying the classic cooperative game-theoretic payoff allocation methods to the emerging multiagent machine learning systems. In the next section, we will introduce the basic framework and some important solution concepts of cooperative games, following the definitions from Chalkiadakis et al. [22].

2.1.2 Cooperative Game Theory

A cooperative game consists of a finite, non-empty set of *players* $N = \{1, \ldots, n\}$. Subsets of N are referred to as *coalitions* — while in everyday usage this term implies a group of agents with some commitment to cooperative action, we emphasise in this thesis it simply refers to a set of players, and we will denote them by C. The *grand coalition* refers to the set of all players N. In this thesis, we study transferable utility (characteristic function) games, where payoffs to a coalition may be freely redistributed among its members, and a *characteristic function* v maps each possible coalition to a real value. The usual interpretation of $v(C)$ is that this is the value that the players S could obtain if they were to cooperate.

Definition 2.1.2.1 (Cooperative Game)**.** *A cooperative game with transferable utility (hereafter a* cooperative game*) is given by a pair $G = (N, v)$, where $N = \{1, \ldots, n\}$ is a finite, non-empty set of players, indexed by i, and $v\colon 2^N \to \mathbb{R}$ is a* characteristic function*, which assigns a real value $v(C)$ to every coalition of players $C \subseteq N$.*

Often in cooperative games, the joint decision-making process is omitted and the characteristic function is assumed to be provided as an oracle. However, in typical machine learning applications, computation or estimation of the value of coalitions is non-trivial and may require training and evaluating machine learning models. To understand a cooperative game, the following example of ice cream purchase game introduced in [22] demonstrates the basic concepts through a scenario where multiple players pool their resources to obtain a collective payoff.

Example 2.1.2.1. *Alice, Bob, Charlie pool their savings in order to purchase ice cream. There are 3 ice cream tub sizes:* 500*ml for* £5*,* 800*ml for* £8*, and* 1000*ml for* £10*. The children each holds the following amount of savings: Alice has £5, Bob has £3 and Charlie has £2.*

This simple example can be captured by a cooperative game $G = (N, v)$, where $N = \{Alice, Bob, Charlie\}$ are the players. The characteristic function $v\colon 2^N \to \mathbb{R}$ evaluates how

much payoff (in *ml*'s of ice cream) a coalition of players can obtain if they cooperate. For instance, $Alice$ on her own can obtain $500ml$, while together with Bob they can afford $800ml$. According to this definition, the values of all possible coalitions can be computed as follows:

$v(\emptyset) = 0$, where $\emptyset$ denotes the empty coalition,

$v(\{Alice\}) = 500, v(\{Bob\}) = 0, v(\{Charlie\}) = 0,$

$v(\{Alice, Bob\}) = 800, v(\{Bob, Charlie\}) = 500, v(\{Alice, Charlie\}) = 500,$

$v(\{Alice, Bob, Charlie\}) = 1000.$

An *outcome* of a cooperative game consists of a *partition* of players into coalitions and a *payoff* vector which distributes the payoff of each coalition back to its members. Typically, an outcome can be evaluated according to two sets of criteria [20]: 1. *fairness*, that is, does the payoff of a player reflect its contribution? To this end, the fairness of common game-theoretic solution concepts are often evaluated according to several fairness axioms which are elaborated in the next section; and 2. *stability*, that is, is a player or a group of players incentivised to switch coalitions for a better payoff? To this end, many solution concepts are developed to prevent the switching behaviours, such as the core [60].

2.1.2.1 Solution Concepts

In this thesis, we consider the cooperative scenarios where all players are assumed to form a grand coalition N, that is, all players cooperate as a single coalition. While this assumption may seem counter-intuitive in a submodular game in terms of stability, i.e., agents can be incentivised to form smaller coalitions. However in a wide range of real-world multiagent ML applications, the grand coalition is formed, as a result of the system being fully-cooperative (e.g., multi-robot teams) and/or the payoff allocation methods are applied to evaluate the importance of the agents (e.g., a set of passive entities such as data or features). Alternatively, the grand coalition can form as a result of mechanisms that are imposed by the ML application to ensure the cooperation among the agents (e.g., federated learning). In all cases, despite the fact that the game-theoretic solution concepts are widely applied to multiagent ML applications, their theoretical aspects in the submodular games are largely overlooked.

In this thesis, we aim to bridge the gap by studying the theoretical aspects of the classic payoff allocation methods in the submodular games. And we focus on the fairness and robustness of

the payoff allocation methods with respect to the grand coalition. With this assumption, we will define a solution concept by a function that assigns a payoff $\varphi_i(N, v) \in \mathbb{R}$ to each player i.

We focus on a wide class of solution concepts called semivalues [32]. Apart from the semivalues, there are useful solution concepts such as the Owen value [82], which is an extension of Shapley value for cooperative games when a particular coalition structure or partition of the set of players is considered in addition; the Core [121], which is the set of feasible allocations that cannot be improved upon by a coalition of the agents. In this thesis, we focus on the semivalues in multiagent ML systems as they exhibit a set of fairness properties (e.g., axioms such as symmetry which will be discussed later in this section). And due to these properties, the semivalues are commonly adopted in multiagent ML applications.

For clarity, we will introduce the definition of semivalues in Chapter 3 and Chapter 4. Before this, let us introduce the concept of *marginal contribution* and some well-known semivalues. Formally, the marginal contribution of player i towards a coalition C of other players is defined as the difference that the player makes before and after joining a coalition, i.e.,

$$\mathcal{MC}_i(C) \coloneqq v(C \cup \{i\}) - v(C). \tag{2.1}$$

The class of solution concepts based on marginal contributions defines the payoff of a player as a weighted sum of its marginal contributions towards coalitions of other players, i.e.,

$$\varphi_i(N, v) = \sum_{C \subseteq N \setminus \{i\}} w_{C,N} \mathcal{MC}_i(C)$$

The Shapley Value [122] is the most common solution concept for payoff allocation. It is defined as the following weighted average of a player i's marginal contributions towards all possible coalitions of other players:

$$\varphi_i^{\text{Shapley}}(N, v) = \sum_{C \subseteq N \setminus \{i\}} \frac{|C|!(|N| - |C| - 1)!}{|N|!} \mathcal{MC}_i(C). \tag{2.2}$$

The Banzhaf Value [8] is a commonly used measure for voting power, which is defined by the average marginal contribution of a player i towards all coalitions of other players:

$$\varphi_i^{\text{Banzhaf}}(N, v) = \frac{1}{2^{|N|-1}} \sum_{C \subseteq N \setminus \{i\}} \mathcal{MC}_i(C). \tag{2.3}$$

The Leave-one-out Value (LOO) is a simple solution concept which assigns to each player its marginal contribution towards the coalition of all other players:

$$\varphi_i^{\text{LOO}} = \mathcal{MC}_i(N \setminus \{i\}).$$

The leave-one-out value is commonly used [42] to compute an agent's contribution towards the group due to its computational simplicity.

Having defined the common solution concepts, we now illustrate how these solution concepts such as the Shapley value can be computed based on the value of coalitions. Recall the ice cream example, given the value of coalitions, we can first compute the marginal contribution of each player. Take *Alice* as an example,[2]

$\mathcal{MC}_{\text{Alice}}(\emptyset) = v(Alice) - v(\emptyset) = 500,$

$\mathcal{MC}_{\text{Alice}}(Bob) = v(Alice, Bob) - v(Bob) = 800,$

$\mathcal{MC}_{\text{Alice}}(Charlie) = v(Alice, Charlie) - v(Charlie) = 500,$

$\mathcal{MC}_{\text{Alice}}(Bob, Charlie) = v(Alice, Bob, Charlie) - v(Bob, Charlie) = 500.$

Then, we compute the payoff of each player as a weighted sum of its marginal contributions:

$$\begin{aligned}\varphi_{\text{Alice}}^{\text{Shapley}} &= \sum_{C \subseteq N \setminus \{\text{Alice}\}} \frac{|C|!(|N| - |C| - 1)!}{|N|!} \mathcal{MC}_{\text{Alice}}(C) = \sum_{C \subseteq N \setminus \{\text{Alice}\}} \frac{1}{|N|} \binom{|N|-1}{|C|}^{-1} \mathcal{MC}_{\text{Alice}}(C) \\ &= \frac{1}{3}\mathcal{MC}_{\text{Alice}}(\emptyset) + \frac{1}{6}\mathcal{MC}_{\text{Alice}}(Bob) + \frac{1}{6}\mathcal{MC}_{\text{Alice}}(Charlie) + \frac{1}{3}\mathcal{MC}_{\text{Alice}}(Bob, Charlie) \\ &= \frac{1}{3} \times 500 + \frac{1}{6} \times 800 + \frac{1}{6} \times 500 + \frac{1}{3} \times 500 = 550.\end{aligned}$$

Similarly we can obtain for the other players $\varphi_{\text{Bob}}^{\text{Shapley}} = 300$, and $\varphi_{\text{Charlie}}^{\text{Shapley}} = 150$.

Having reviewed some common solution concepts, an immediate question that follows is how to evaluate how good (or bad) a solution concept is for a specific application. Moreover, provided a set of different solution concepts, which one is the most suitable for the application? For instance, in Example 2.1.2.1, a simple solution concept is for the children to purchase and divide the 1000ml ice cream equally. In that case, however, *Alice* may argue that the payoff allocation is unfair as she contributed the most while receiving a payoff equal to the other players who contributed less. To answer this question, solution concepts are commonly evaluated based on a collection of natural axioms such as the following [22]:

[2]Occasionally, for the clarity of presentation, we denote $v(\{i, j\})$ as $v(i, j)$, and overload the notation N as both the grand coalition, and the cardinality of the grand coalition.

(A1) *Symmetry*: Two players i and j who have the same marginal contribution in any coalition have the same payoff, i.e., $(\forall S \subseteq N \setminus \{i,j\} : v(S \cup \{i\}) = v(S \cup \{j\})) \rightarrow \varphi_i(N,v) = \varphi_j(N,v)$.

(A2) *Efficiency*: The payoffs to all players sum to $v(N)$, i.e., $v(N) = \sum_{i \in N} \varphi_i(N,v)$.

(A3) *Null-player*: a player whose marginal contribution is zero in any coalition has zero payoff, i.e., $(\forall S \subseteq N : v(S \cup \{i\}) = v(S)) \rightarrow \varphi_i(N,v) = 0$.

(A4) *Linearity*: Given two cooperative games $G^1 = (N, v^1)$ and $G^2 = (N, v^2)$, then for any player $i \in N$, $\varphi_i(N, v^1 + v^2) = \varphi_i(N, v^1) + \varphi_i(N, v^2)$.

(A5) *2-Efficiency* [72]: $\varphi_i(N,v) + \varphi_j(N,v) = \varphi_{p_{ij}}(N',v')$ characterises neutrality of collusion, where $\varphi_{p_{ij}}(N',v')$ is player p_{ij}'s payoff in a game in which players i and j merged as a single player p_{ij}, i.e., $N' = N \setminus \{i,j\} \cup \{p_{ij}\}$.

The Shapley value is the unique solution concept that satisfies Axioms **(A1)**-**(A4)**. It is commonly considered a *fair* payoff allocation method as the total payoff of the system is distributed among the players, players with equal contributions receive equal payoffs, and a null player receives zero payoff. On the other hand, the Banzhaf value is uniquely characterized by Axioms **(A1)**, **(A3)**-**(A5)**, which replaces the efficiency property with the 2-efficiency property.

2.1.3 Malicious Manipulations

In many cases, players in a cooperative multiagent system are still rational (self-interested). This can sometimes give rise to malicious behaviours performed by the players in order to optimise their own payoffs. We will demonstrate in Chapter 3 that in such scenarios, the robustness properties of solution concepts against manipulations are important apart from their fairness properties.

Several seminal works have studied behaviours of solution concepts against manipulations, such as merging (and splitting) proofness of solution concepts, collusion, and false name manipulations in the cooperative game theory literature. For example, Lehrer [72] first presented an axiomatization of the Banzhaf value with the 2-efficiency property, which characterized the neutrality of the Banzhaf value against two players merging as one. This malicious manipulation is captured by *amalgamation* as the following game.

Definition 2.1.3.1 (Amalgamation Game)**.** *Let $G = (N, v)$ be a cooperative game. Let $i, j \in N$ be two players. By merging i, j as a single player $m_{i,j}$, an* amalgamation game *$G' = (N', v')$ is induced, where $N' = \{m_{i,j}\} \cup N \setminus \{i, j\}$, and the new characteristic function is given by $\forall C \subseteq N'$,*

$$v'(C) = \begin{cases} v(C) & \text{if } m_{i,j} \notin C \\ v(C \setminus \{m_{i,j}\} \cup \{i, j\}) & \text{otherwise.} \end{cases}$$

The process in reverse to amalgamation (merging) of the two players is splitting, that is, a player $m_{i,j}$ in the amalgamated game can be split into two players i, j. In this context, G' is the original game and G now becomes the induced game due to splitting. The 2-efficiency axiom **(A5)** satisfied by the Banzhaf value, states the neutrality of both splitting and merging: the payoff of the merged player $m_{i,j}$ in the amalgamated game G' is equal to the sum of two merging players' individual payoffs in the original game G, i.e., $\varphi_{m_{i,j}}(N', v') = \varphi_i(N, v) + \varphi_j(N, v)$. We will also demonstrate in later chapters that the Banzhaf value is also robust against another type of manipulation relevant to many machine learning applications. Apart from Lehrer [72], van den Brink [141] studied the interplay between efficiency and collusion neutrality of two players, and states that no solution concepts satisfy efficiency, collusion neutrality and the null player property at the same time. Ohta et al. [103] proposed anonymity-proof Shapley value against malicious players who split their resources (skills) and act as multiple false identities.

2.2 Reinforcement Learning (RL)

In this section, we introduce the background for reinforcement learning (RL).

2.2.1 Elements of Reinforcement Learning

In RL, an *agent* interacts with an *environment* and learns to perform tasks through trial and error. Typically, the decision making of an RL agent is modelled by a Markov Decision Process (MDP), and the agent's objective is commonly conveyed through a reward function that evaluates its performance [133]. During training, the goal of the agent is to learn an optimal *policy* which maximizes the cumulative future rewards.

2.2.1.1 Markov Decision Process (MDP)

A *Markov Decision Process (MDP)* is commonly used to model an agent's decision making process: $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle$. At each time step t, an agent in state $s_t \in \mathcal{S}$, chooses an action $a_t \in \mathcal{A}$, receives a reward $r_t = r(s_t, a_t)$, and transits to the next state $s_{t+1} \in \mathcal{S}$ according to an underlying transition dynamics which characterises the environment $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$. The transition dynamics and reward function are often assumed to be unknown to the agent.

The agent aims to maximise the discounted future *return* $G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$, where $\gamma \in [0, 1]$ is a discount factor. Intuitively, the discounted future return refers to the discounted cumulative future rewards the agent can obtain, where the discount factor favours the nearby timesteps. While the agent interacts with the environment, it collects and stores experiences of transition tuples and updates its policy from them. We denote an agent's transition tuple at each step as $\mathcal{D}_t = \langle s_t, a_t, s_{t+1}, r_t \rangle$.

The *policy* of an agent is a mapping from states to probabilities of selecting a possible action, denoted as $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$. During training, the goal of the agent is to update the policy in order to maximise the total expected discounted return per episode $\mathbb{E}_\pi[G_0]$.

2.2.1.2 Value Functions and Bellman Equations

The *state-value function*, denoted $V^\pi(s)$, is the expected return of an agent which starts in s and follows policy π thereafter:

$$V^\pi(s) \coloneqq \mathbb{E}_\pi[G_t | s_t = s] = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t = s], \forall s \in \mathcal{S}. \tag{2.4}$$

The *action-value function* $Q^\pi(s, a)$ defines the expected return when starting from s, taking action a, and following π thereafter:

$$\begin{aligned} Q^\pi(s, a) &\coloneqq \mathbb{E}_\pi[G_t | s_t = s, a_t = a] \\ &= \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t = s, a_t = a], \forall s \in \mathcal{S}. \end{aligned} \tag{2.5}$$

The state and action value functions are related as follows:

$$V^\pi(s) = \sum_a \pi(a|s) Q^\pi(s, a), \qquad Q^\pi(s, a) = r_s^a + \gamma \sum_{s'} P_{s,s'}^a V^\pi(s') \tag{2.6}$$

The *Bellman equation* expresses the recursive relation of the value functions, i.e.

$$V^{\pi}(s) = \sum_{a} \pi(a|s) \left(r_s^a + \gamma \sum_{s'} P_{s,s'}^a V^{\pi}(s') \right), \qquad Q^{\pi}(s,a) = r_s^a + \gamma \sum_{s',a'} P_{s,s'}^a \pi(a'|s') Q^{\pi}(s',a'), \tag{2.7}$$

where $P_{s,s'}^a$ is the probability of transiting to state s' after taking action a in state s, and $\pi(a|s)$ is the probability of the agent taking action a in state s.

2.2.2 Model-free Reinforcement Learning

Having defined the basic elements of RL, we now briefly introduce how the agents learn to update their policies. RL methods can be broadly categorised into model-free and model-based RL. Model-free RL algorithms specify how agents learn to update their policies to optimise the task performance, while model-based RL agents additionally fit and utilise an estimated model of the environment. We will return to model-based RL in Section 2.2.5.

Model-free methods [133] are categorized into *value-based* algorithms and *policy-based* algorithms. Specifically, value-based algorithms learn to estimate the value functions and derive a policy implicitly, while policy-based algorithms explicitly learn the policy. An important class of value-based algorithms use Generalised Policy Iteration (GPI), which refers to an iterative procedure of policy evaluation and policy improvement. During policy evaluation, the agent estimates the state or action-value function $Q^{\pi}(s,a)$ of the current policy π. Then during policy improvement, the agent generates a policy by acting ϵ-greedily according to the value function $\pi'(s) = \arg\max_{a \in \mathcal{A}} Q^{\pi}(s,a)$. This procedure is then repeated till convergence to approach optimality.

Several methods can be used for value estimation, such as Monte-Carlo learning or *Temporal Difference (TD)* learning. With Monte-Carlo methods, we can update the value function estimation at each state s through the empirical future return G_t computed after the end of each episode, for example, to update the estimated state value function,

$$V(s) \leftarrow V(s) + \alpha[\frac{\sum_t \mathbb{1}_{s_t=s} G_t}{\sum_t \mathbb{1}_{s_t=s}} - V(s)],$$

where α is a parameter for learning rate, and $\mathbb{1}$ is the indicator function.

Whereas with TD methods, we do not need to wait until the end of each episode in order to perform an update. Instead, we can perform an update after the next timestep, and the value

function estimation can be updated towards a TD target $\hat{G}_t^1$, or 1-step return:

$$V(s_t) \leftarrow V(s_t) + \alpha[\overbrace{\underbrace{r_t + \gamma V(s_{t+1})}_{\text{TD error } \delta_t} - V(s_t)}^{\text{1-step return } \hat{G}_t^1}].$$

The term $\delta_t \coloneqq r_t + \gamma V(s_{t+1}) - V(s_t)$ is called the TD error, which measures the difference between the estimated value of state s_t and the TD target $r_t + \gamma V(s_{t+1})$, which is a better estimate of state value of s_t. As a unification of the Monte-Carlo and (one-step) TD target, we can define the n-step return as:

$$\hat{G}_t^n \coloneqq r_t + \gamma r_{t+1} + \ldots + \gamma^{n-1} r_{t+n-1} + \gamma^n V(s_{t+n}) \tag{2.8}$$

Algorithm 1 On-policy TD Control Algorithm (SARSA)[133]

Input: exploration parameter ϵ, step-size α
Output: (approximate) optimal action values $Q(s, a)$
1: Loop for each episode:
2: Initialize starting state s_0
3: Choose action a_0 by policy via $Q(s_0, a)$, (e.g., ϵ-greedy with $\pi(a|s_0) = \arg\max_a Q(s_0, a)$)
4: Loop for each step $t = 0, 1, \ldots$
5: Take action a_t, observe reward and next state r_t, s_{t+1}
6: Choose action a_{t+1} from s_{t+1} via policy derived from $Q(s_{t+1}, a)$, (e.g., ϵ-greedy)
7: $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_t + \gamma Q(s_{t+1}, a) - Q(s_t, a_t)$.
8: until S_t is terminal

Algorithm 1 shows an example value-based algorithm, i.e., On-policy TD control (SARSA) [133], which learns to update the action value estimation through TD learning and perform policy improvement by acting ϵ-greedily according to $Q(s, a)$. Note that the update step (line 7) is the one-step TD update.

Unlike value-based methods, policy-based methods explicitly learn and update the policy π of the agents. A common class of policy-based algorithms called *actor-critic methods* learn a value function in addition to the policy function. Specifically, an *actor* corresponds to the policy, while a *critic* corresponds to a value function. During training, the actor update is guided by the critic using the Policy Gradient Theorem [133]. During execution, only the actor (the policy) is necessary for the agent to operate by choosing an action at each step. while the critic is only needed during training. As we will discuss in Section 2.2.3 and 2.2.4, the actor-critic learning framework is important in both single-agent and multiagent learning.

2.2.3 Reinforcement Learning for Continuous Control

Many real-world problems exhibit a continuous action space. For example, for many robotic control problems, the output actions typically correspond to joint torques or forces which are continuous [68]. Moreover, the observations from the sensors are often high-dimensional and continuous features, resulting in both continuous state and action spaces.

To estimate the value functions (i.e., $Q^\pi(s,a)$ or $V^\pi(s)$) with continuous state and action spaces, we can use function approximation, e.g., by using a neural network. Typically in continuous spaces, using policy-based methods are a more natural choice, because there is an infinite number of states and actions and to estimate the values in continuous problems and hence value-based approaches can be computationally expensive [146]. To model a policy with continuous action spaces, each dimension of the output action can be defined by a Gaussian distribution over a real-valued scalar action, with mean and standard deviation given by a neural network output that depends on the state. Alternative distributions may include the Beta distribution [25] for bounded action spaces or skewed distribution with application-specific knowledge. In this thesis we adopt the Gaussian distribution as it is a standard and most widely adopted distribution for continuous control tasks [133]. There are several policy-based algorithms capable of performing continuous control, such as deep deterministic policy gradient (DDPG) [77], trust region policy gradient (TRPO) [116], proximal policy gradient (PPO) [118]. Next, we will introduce Proximal Policy Gradient [118], a standard algorithm that is commonly adopted for continuous control, which will be used in this thesis for robotic control problems.

2.2.3.1 Proximal Policy Optimisation (PPO)

Proximal Policy Gradient (PPO) [118] is an actor-critic policy gradient RL algorithm that is widely used in both discrete and continuous control tasks, due to its simplicity in implementation, sample complexity, and ease of hyperparameter tuning. The pseudocode for PPO is shown in Algorithm 2. Specifically, the *actor* corresponds to the policy $\pi(a|s)$, while the *critic* corresponds to the value function $V^\pi(s)$. The actor update is guided by an *advantage function* $A(s,a)$ estimated using the critic, which is defined as the following equation:

$$A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s) \tag{2.9}$$

Algorithm 2 Proximal Policy Gradient Algorithm (PPO)

1: **Initialise:** Critic network parameters V_ω, and Actor parameters π_θ.
2: **for** iterations $k = 0, 1, 2, \ldots$ **do**
3: Collect sets of trajectories $\mathcal{D}_k = \{\tau\}$ by running policy π_k in the environment.
4: Compute empirical returns G_t
5: Compute the advantage function $A(s_t, \mathrm{a_t})$ according to the critic,
6: Update the actor by maximising the PPO-Clip objective (in minibatches):
7: $\theta_{k+1} = \arg\max_\theta \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min\Big(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A(s_t, \mathrm{a_t}), \mathrm{clip}\big(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}, 1-\epsilon, 1+\epsilon\big) A(s_t, a_t) \Big)$,
8: # Update the critic by regression with mean squared error:
9: $\omega_{k+1} = \arg\min_\omega \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \big(V_\omega(s_t) - G_t\big)^2$
10: **end for**

The advantage function of a state-action pair, defined by subtracting a baseline state-value function from the action-value function, measures how much better (worse) taking action a in state s is compared with the policy's default behaviour. Intuitively, if the advantage $A(s, a)$ is positive, the actor parameters move towards the direction where this action becomes more likely. To improve stability, PPO avoids parameter updates that change the policy drastically within each iteration, by constraining the ratio between old and new policies $r(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}$ to be between $[1-\epsilon, 1+\epsilon]$. This is achieved by clipping, i.e., $\mathrm{clip}\big(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)}, 1-\epsilon, 1+\epsilon\big) A(s_t, a_t)$, c.f. Line 7 of Algorithm 2.

Though defined as $A(s_t, a_t) = Q(s_t, a_t) - V(s_t)$, in practice, we do not need two separate critics for estimating both the state and action value functions. Alternatively, $A^\pi(s, a)$ can be estimated using methods such as general advantage estimation (GAE) [117], which makes use of the state value function $V^\pi(s)$ and the n-step returns. Specifically, the TD error $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$ which we discussed in the previous section can be used to estimate the advantage function. Alternatively, we can have advantage estimators using the n-step returns, i.e.,

$$\hat{A}_t^n = \hat{G}_t^n - V(s_t) = r_t + \gamma r_{t+1} + \ldots + \gamma^{n-1} r_{t+n-1} + \gamma^n V(s_{t+n}) - V(s_t)$$

Then the generalised advantage estimator can be defined by a λ exponentially-weighted average over the n-step advantage estimators,

$$\hat{A}^{\mathrm{GAE}(\gamma,\lambda)} = (1-\lambda)(\hat{A}_t^1 + \lambda \hat{A}_t^2 + \lambda^3 \hat{A}_t^3 + \ldots),$$

The detailed derivation of the generalised advantage functions can be found in [117].

2.2.4 Multiagent Reinforcement Learning

In this section, we will discuss some common multiagent RL algorithms. We will first introduce the *Markov Game*, which is a common framework for modelling multiagent decision making.

A *Markov Game* [80] is a tuple $(N, \mathcal{S}, P, \mathcal{A}, r)$ where N is the set of n players $\{1, 2, ..., n\}$, $\mathcal{S}$ is a set of states of the environment, $\mathcal{A} = A_1 \times A_2 ... \times A_n$ is the set of joint actions where A_i is the set of actions available to agent i. $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}^n$ is the reward function which maps a state and joint action to a reward for each agent. $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is the transition probability of transitioning from state s to s' under a joint action.

Many algorithms have been developed for multiagent RL. In the most simple form, a *Joint Action Learner* [43] is a single central controller which simultaneously chooses a joint action for all agents. In contrast, *Independent Q-learning (IQL)* [139] is a framework where each agent regards the other agents as part of the environment. This provides a decentralized system that usually performs surprisingly well in multiagent tasks. Several methods extend (non-cooperative) game-theoretic solution concepts to multiagent RL, for example, *Correlated Q Learning* [47], *minimax-Q Learning* [80], *Nash-Q Learning* [59], etc. The essential idea of this group of algorithms is that they replace the greedy action selection with actions that form an equilibrium, such that the agents converge to a specific equilibrium solution. In terms of training, *Self-play* [125] is a common paradigm which effectively reduces the number of parameters for training a multiagent RL system. As the name suggests, an agent plays with or against itself. Only a single agent model is learned, and each agent is an instantiation of the model. The shared model is updated using the experiences of all agents' in the system. More related works in multiagent RL have been discussed in the related work section (Section 4.2) in Chapter 4.

2.2.4.1 Centralised Training, Decentralised Execution (CTDE)

Recently, a standard practice in training a decentralised multiagent RL system is Centralised Training, Decentralised Execution (CTDE) [42]. For example, in a multiagent system with agents $N = \{1, \ldots, n\}$, a *centralised critic* – typically a centralised action-value functions $Q(\mathrm{s}_t, a_1, \ldots, a_n)$), and decentralised actors $(\pi_1, \ldots, \pi_n)$ – the policies of the agents, are modeled. The centralised critic takes as input the observations and joint actions of all agents, while the decentralised actors take the agents' local observations as input and output an action per agent.

Compared with having decentralised critics (one per agent) which only condition on each agent's local observation and action, the advantage of CTDE is that the centralised critic prevents an agent from experiencing a non-stationary environment. The problem of non-stationarity arises as a result of the agent treating other agents who are simultaneously learning as part of the environment. Moreover, the benefit of actor-critic methods is that the critic is only used during training, while during execution only the decentralised actors are used. On the one hand, the agents learn coordinated behaviours during training with the help of the critic. On the other hand, the coordinated policies can be performed by the agents in a decentralised manner during execution. In the later chapters, the CTDE framework will be utilised for our credit assignment framework for multiagent RL.

2.2.5 Model-based Reinforcement Learning (MBRL)

So far we have mainly discussed model-free RL methods which do not explicitly utilise a model of the environment. In this section, we will give a brief introduction to model-based RL methods. Unlike model-free methods where agents focus on optimising the returns and only implicitly learn about the environment model, model-based RL additionally allows agents to learn and utilise a model of the environment. Having an environment model provides many benefits. For example, agents can make more informed predictions of an outcome of an action [36]. Moreover, agents can reason about the environment, such as uncertainty and safety analysis, which are crucial in safety-critical domains such as robotics and autonomous driving [40, 112].

Typically in model-based RL, the agent learns an estimated *model* which can encode knowledge of the environment. For example, a simple model of the transition dynamics can be defined as

$$\hat{s}_{t+1} = f_s(s_t, a_t), \qquad \text{(Transition Model)}$$

where $f_s : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ maps a state-action pair to a predicted next state. For problems with high-dimensional observations such as images, this dynamics model can be defined in a latent state space, then the estimated next state can be used to reconstruct the observations [49]. Similarly, a model of the reward function can be defined as

$$\hat{r}_t = f_r(s_t, a_t), \qquad \text{(Reward Model)}$$

where $f_r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ maps a state-action pair to a real-valued reward.

Algorithm 3 Dyna Algorithm (Tabular) 3

Initialise: $Q(s, a)$ and Model $f = (f_s, f_r)$
Loop for each step $t = 0, 1, \ldots$
  $s \leftarrow$ current (non-terminal) state
  Choose action a by ϵ-greedy policy according to $Q(s, a)$
  Take action a; observe resultant reward r, and state s'
  Update by Q-learning: $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a^*} Q(s', a^*) - Q(s, a)]$
  Update transition and rewards model $f_s(s, a) \leftarrow s', f_r(s, a) \leftarrow r$
  Loop repeat for n times: # Agent update using model simulations
    $s \leftarrow$ random previously observed state, $a \leftarrow$ random action previously taken in S
    simulate next state and reward using learned model $s' \leftarrow f_s(s, a), r \leftarrow f_r(s, a)$
    Update using simulated experience: $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a^*} Q(s', a^*) - Q(s, a)]$

The model can be used in various ways, for example, decision-time planning [133, 52, 125] which uses simulated experiences to choose actions during execution. During training, the model can also be used to improve an agent's value and policy learning, by generating simulated experiences that augment the agent's experiences collected from real-world interactions. Examples include model-based value expansion (MVE) [37], model-based policy optimization (MBPO) [61], Dyna [132], which iterates between model fitting and updates of the agent policies and value function estimations in each iteration of learning. The steps of a simple tabular Dyna algorithm with the Q-learning agent are shown in Algorithm 3.

This concludes the background section, where we introduced the common concepts in cooperative game theory and RL for understanding the rest of the thesis. More specific background concepts and related works are introduced within each chapter.

3

Semivalues in Cooperative Games with Submodular Characteristic Functions

Contents

3.1 Introduction

In this chapter, we study game-theoretic payoff allocation methods in multiagent ML systems exhibiting submodular properties. Submodular functions are an important class of set functions studied in optimisation problems that satisfy the diminishing returns property. Informally, the marginal value that an element makes when added to a set decreases as the set grows bigger. This property frequently occurs in real-world settings such as economics and engineering, making submodular functions a powerful mathematical model for a wide range of applications. A daily example is the cost or utility of buying items such as coffee, where purchasing an increased amount of items often lead to larger discounts and diminishing marginal utilities. Generally, in minimisation problems, submodular functions can often model notions of *similarity (irregularity)* and *cooperative costs*, while in maximisation problems, submodularity can often model notions of *diversity*, *information* and *coverage*. Classic examples include sensor placement problems for monitoring spatial information [69], and facility location problems [28]. Nowadays, submodular functions have taken an increasingly important part in the field of machine learning. For example, a standard submodular ML problem is data summarisation [11, 78]: typically data carry *redundant information* and it is useful to find a concise subset as the representation of the data which reduces the redundancy while maintaining the level of diversity. In this case, the diversity of the data can be modelled by a submodular function and the data summarisation problem can then be cast as a constrained submodular optimisation problem.

In cooperative game theory, *convex games* are commonly studied due to their attractive properties such as the existence of non-empty core [121]. However, as we discussed above and in Chapter 1, many real-world multiagent ML applications exhibit submodular behaviours due to natural characteristics of the tasks and resources, such as overlapping information in ML data and ML features. Moreover, the properties of ML applications give rise to the increased computational burden, as well as a new type of malicious attack, namely, replication manipulation. In this chapter, we look into the properties and aforementioned challenges in submodular cooperative games and apply our theoretical results to an emerging multiagent ML application. Inspired by the wide use of Shapley value in multiagent ML systems [63, 75, 84**?**], we extend from the Shapley value and study a family of Shapley-like solution concepts called the Semivalues [32].

In Section 3.3, we study the theoretical aspects of semivalues in submodular games and their behaviours under replication manipulations. An outline is as follows:

Firstly, we define submodular games and provide a motivating example of the submodular facility location problem (FLP). In the facility location game, we derived closed-form solutions which give rise to efficient algorithms for computing the two most common semivalues, namely, the Shapley value and the Banzhaf value. In the later sections, these algorithms are also used to empirically validate our results on large scale experiments.

Secondly, we study the replication manipulation in general submodular games, where a player replicates itself (e.g., its resource) and acts under multiple identities. Here, we make an assumption that the replicated identities do not add additional values towards other replicas. Inspired by the classic literature on splitting and merging manipulations, we first investigate the case where a player replicates itself once and acts as two identities (i.e., $k = 1$ number of replications). We then show that the Shapley value and the Banzhaf value display distinct behaviours in terms of the total payoff of the malicious player, by building a matching between coalitions and applying the submodularity property.

Following this result, we extend our investigation into the general case where the malicious player replicates itself an arbitrary number of times. However, we observe that it is difficult to extend the previous coalition matching proof towards the case of general number of replications, (i.e., $k \geq 1$). To solve this problem, we found an alternative solution by representing the semivalues as a weighted sum over a player's marginal contributions towards different coalition sizes. In particular, we show that for submodular games, the average marginal contribution of a player monotonically decreases with coalition sizes. Based on this property, we present a necessary and sufficient condition which characterises the robustness of all semivalues under replication. For example, we show that with the Shapley value, the total payoff of the malicious player monotonically increases with the number of replications, and converges to the player's characteristic value.

In Section 3.4, we extend our theoretical study to an emerging application, namely, multiagent ML data markets. In particular, we envision a data market that connects data providers with ML practitioners who are in need of application-specific and up-to-date data. To train an ML model, data from multiple data providers are uploaded to a secure cloud platform, and a model is trained using their collective data. We model the data market as a submodular

cooperative game where the players (i.e., the data providers) receive payoffs according to their data's contributions towards the ML model performance. In order to gain a higher payoff, a player is incentivized to replicate its data and act under multiple false identities. We will first discuss the application of our theoretical results from Section 3.3 to the data market. Then, extending from our replication model, we will discuss some related attack methods such as a combination of data replication and perturbation, data splitting, or replicating subsets of the data. Finally, to address the expensive computation in the data market payoff allocation, we introduce a sampling algorithm for estimating the semivalues, which demonstrates strong empirical performances against the state-of-the-art sampling algorithms.

In Section 3.5 we empirically validate our theoretical results on experiments using the facility location game and experiments using real-world machine learning datasets.

3.2 Related Work

Our theoretical results relate closely to the seminal game-theoretic literature on merging and splitting proofness, collusion, and false name manipulations. [72] is the first to present an axiomatization of the Banzhaf value with the 2-efficiency axiom, which characterized the neutrality of the Banzhaf value on merging two players as one. Alternatively, [51] studied the collusion of two players where they both keep their identities thus the total number of players is unchanged: under a proxy agreement, one player acts as a proxy while the other a null player, whereas under an association agreement, two players act on each other's behalf. [141] studied the interplay between efficiency and collusion neutrality of two players, and state that no solution concepts satisfy efficiency, collusion neutrality and the null player property at the same time. [67] studied the merging and splitting-proofness on the class of convex games, and introduced some possibility/impossibility results, for instance, no solution concept satisfies anonymity and splitting-proofness on the class of monotonic convex games. Our present work looks at the setting of novel digital resources (such as data, ML features) which can typically be replicated at a low cost and exhibit submodular properties. Such cases have not been adequately addressed previously. Moreover, we considered an arbitrary number of replications, beyond the study of bilateral collusion or amalgamation. In addition, our proposed replication-robustness condition is

not only an existence result but a necessary and sufficient condition that can be used to characterize a wide range of solution concepts and can be useful for building new payoff allocation schemes.

The topic of digital resources valuation such as data has drawn recent research interest from a wide range of communities such as economics, machine learning and data science [105, 45, 24, 76, 151, 92, 115, 152, 119]. As digital resources such as data are often invaluable [105], it is important to develop methods for evaluating the contribution of the resources to a task, as well as fair allocation of the payoffs [98]. Fortunately, fair payoff allocation has been extensively studied in the cooperative game theory literature [22, 148], the most common payoff division scheme being the Shapley value [120, 111], which possesses four desirable fairness properties: symmetry, efficiency, null-player and linearity. However, as we will show in the context of digital resources, these fairness properties may not be sufficient and new properties such as robustness against manipulations need to be investigated. Moreover, our theoretical results have implications for game-theoretic payoff allocation in general ML applications, which apply to digital resources such as ML features: A recent line of work in the interpretable ML literature [91] studies ML feature valuation [130, 62], which aims to explain prediction models through the contributions of features. Many have adopted cooperative game theory for valuating the contribution of features, especially the Shapley Value [131, 85, 27, 23], most notable among these work is the SHAP (SHapley Additive exPlanations) framework proposed by [84].

The concept of an ML data market has been explored in the literature. For instance, [2] introduced an algorithmic framework for data markets and addressed issues arising from freely replication of data. To overcome this problem, they propose to use similarity metrics for replica detection. [102] addresses the data replication problem in multiagent ML systems through building a collaborative market where each player must participate both as seller and buyer. This naturally discourages replication, and may apply to alternative application settings. Regarding false name manipulation [5] in an open environment, [103] proposed anonymity-proof Shapley value against malicious players who split their *skills* [7] and act as multiple identities. Their solution to this problem rely on an assumption that each skill is unique and cannot appear under multiple identities.

Efficient computation of the Shapley value has been studied in various settings, e.g., random sampling in [18] and [35] for weighted voting games. Later, stratified sampling was proposed such as in [86, 19]. Most relevant to our work are [46, 63] on ML data valuations. However,

both [46, 63] tend to focus on approximating values of single training datapoints. The former performs approximations during training, which may not represent the Shapley value without the convergence of the training algorithm at each point. The latter proposed a series of algorithms including group testing, as we described and tested empirically, while other approaches rely on properties of single datapoints.

3.3 Theoretical Properties of Semivalues in Submodular Games

Submodularity is commonly used to approximate the behaviours of many real-world applications — the marginal value of resources typically diminishes with the growing amount of resources [66]. In this chapter, we study the properties of solution concepts in cooperative games with submodular characteristic functions.

3.3.1 Submodular Games

In this section, we first introduce submodular functions. Then we define submodular cooperative games and present some properties of their characteristic functions. In particular, we will focus on characteristic function games, and hereafter will refer to them as cooperative games.

The next property lists three equivalent definitions of submodular functions:

Assumption 3.3.1.1 (Submodular Set Functions)**.** *Let N be a finite set, a submodular function is a set function $f : 2^N \to \mathbb{R}$, where 2^N denotes the power set of N , which satisfies one of the following equivalent conditions:*

- $\forall X, Y \subseteq N$ *with* $X \subseteq Y$ *and* $\forall x \in N \setminus Y$*, we have* $f(X \cup \{x\}) - f(X) \geq f(Y \cup \{x\}) - f(Y)$.
- $\forall S, T \subseteq N$*, we have* $f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$.
- $\forall X \subseteq N$ *and* $x_1, x_2 \in N \backslash X$ *such that* $x_1 \neq x_2$*, we have* $f(X \cup \{x_1\}) + f(X \cup \{x_2\}) \geq f(X \cup \{x_1, x_2\}) + f(X)$.

Due to the natural relation between submodular set functions and marginal contributions in cooperative games, we will define submodularity in cooperative games using the first one of the equivalent conditions.

Definition 3.3.1.1 (Submodular Game)**.** *A characteristic function game $G = (N, v)$ with a finite non-empty set of players $N = \{1, \ldots, n\}$, is a* submodular game *if the characteristic function v is submodular, i.e, $\forall C \subseteq C' \subseteq N \setminus \{i\} : v(C \cup \{i\}) - v(C) \geq v(C' \cup \{i\}) - v(C')$.*

In this definition, the difference made by a player i by joining a coalition C is denoted as the *marginal contribution* of player i towards coalition C, i.e., $\mathcal{MC}_i(C) \coloneqq v(C \cup \{i\}) - v(C)$. As a result of submodularity, the marginal contribution of a player towards a coalition C is no less than its contribution towards a superset C', as summarised in the next corollary.

Corollary 3.3.1.1. *In a submodular game $G = (N, v)$, the marginal contributions of each player $i \in N$ satisfy the property:*

$$\forall C \subseteq C' \subseteq N \setminus \{i\} : \mathcal{MC}_i(C) \geq \mathcal{MC}_i(C'). \tag{3.1}$$

Many other studies on cooperative games often focus on convex games where the characteristic function is supermodular. This is because in a convex game (which is necessarily superadditive, i.e., $v(C_1 \cup C_2) \geq v(C_1) + v(C_2)$), players are naturally incentivized to cooperate and form the grand coalition. Moreover, in a convex game, a player is more useful in terms of marginal contribution when it joins a bigger coalition [22]. However, many multiagent ML applications are submodular or approximately submodular, where the marginal contributions of resources diminish with increasing amounts. Though under the submodularity assumption, the players may be incentivized to form smaller coalitions, in many ML applications, the grand coalition will often form due to the requirements of the specific application, agreements among the players, or the players being fully-cooperative. Recall the example scenario where multiple hospitals collaboratively train a joint ML prediction model by securely pooling their medical images, the hospitals will agree to cooperate and form the grand coalition in order to train a better prediction model, even though a player may be less useful with increasing resources.

Fortunately, common payoff allocation solution concepts such as the Shapley value are defined with respect to the grand coalition and are well-defined even in games that are not superadditive. In the rest of the thesis, we will mainly focus on the robustness properties of payoff allocation methods with respect to the grand coalition.

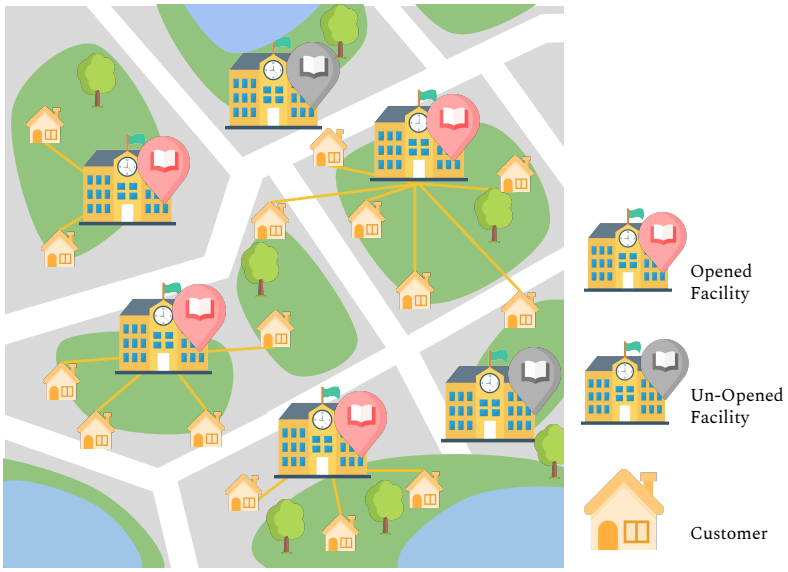


Figure 3.1: An example facility location scenario. The icons are designed by Freepik [44]

3.3.2 Motivating Example: Facility Location Problem

In this section, we look at a classic example of submodular functions, i.e., the facility location problem (FLP). As an important topic in operations research, an FLP considers the question of how to select a cost-effective subset from a given set of potential locations for placing new facilities [11, 113, 28, 41]. Here, the facilities can refer to hospitals, plants, docking stations, etc. Figure 3.1 illustrates an example scenario where there is a set of potential locations in the city to open new schools which can service the existing households. Once opened, the children in the households can then use their nearest schools. In submodular optimisation, the goal is often to pick a subset of facilities to open, in order to maximise the sum of utilities from each customer. As a motivating example, here we evaluate the importance of each facility location according to two important semivalues, i.e., the Shapley value and the Banzhaf value. By modelling the FLP as a submodular game, we present an efficient algorithm for computing the two solution concepts.

3.3.2.1 FLP as a Submodular Game

There exist several different formulations of the FLP [96, 28]. To conform with our assumptions in the later sections (in particular, Assumption 3.3.3.1), we will adopt the formulation of an un-capacitated facility location problem following Nemhauser et al. [96]. In this formulation, the FLP consists of a set of potential facility sites where new facilities can be opened, and a set of customers that must be serviced. The FLP is un-capacitated when it is always optimal

to satisfy the demand of the customers from the open facility which provides them with the highest utilities e.g., proximity to the facility.

Definition 3.3.2.1 (Facility Location Function)**.** *Let D be a set of customers and $\mathcal{L}$ a set of facility locations. Define a utility function $u : \mathcal{L} \times D \to \mathbb{R}_+$, represented by a utility matrix $U \in \mathbb{R}_+^{|\mathcal{L}| \times d}$, where each entry $u_{id} \in U$ is the utility of customer d for facility location i. Let $C \subseteq \mathcal{L}$ denote a coalition of facility locations. The facility location function is defined by $Fac(C) = \sum_{d \in D} \max_{i \in C} u_{id}$.*

To model the FLP as a cooperative game, consider the set of facility locations $\mathcal{L}$ as the players, and the facility location function as the characteristic function.

Definition 3.3.2.2 (Facility Location Game)**.** *Let $\mathcal{L}$ be a set of facility locations and D be a set of customers with their utility functions $u : \mathcal{L} \times D \to \mathbb{R}_+$, represented by a utility matrix $U \in \mathbb{R}_+^{|\mathcal{L}| \times d}$, where each entry $u_{id} \in U$ is the utility of customer $d \in D$ for facility location $i \in \mathcal{L}$. We define a facility location game as $G = (\mathcal{L}, v)$, where $\mathcal{L}$ is the set of players and the characteristic function is the facility location function $v(C) = Fac(C) = \sum_{d \in D} \max_{i \in C} u_{id}$.*

The facility location function has been shown to be submodular [11] in prior work. Here we provide a brief proof sketch in the context of the facility location game.

Proof Sketch for Submodularity. We aim to show that $\forall i \in \mathcal{L}, C_1 \subseteq C_2 \subseteq \mathcal{L} \setminus \{i\}, \mathcal{MC}_i(C_1) \geq \mathcal{MC}_i(C_2)$. By definition, $\mathcal{MC}_i(C_1) = v(C_1 \cup \{i\}) - v(C_1) = \sum_d \max_{j \in C_1 \cup \{i\}} u_{jd} - \sum_d \max_{j \in C_1} u_{jd}$, and $\mathcal{MC}_i(C_2) = v(C_2 \cup \{i\}) - v(C_2) = \sum_d \max_{j \in C_2 \cup \{i\}} u_{jd} - \sum_d \max_{j \in C_2} u_{jd}$. We look at $\mathcal{MC}_i^d(C_1) - \mathcal{MC}_i^d(C_2)$ for each dimension (customer) d. There are three cases:

(i) location i yields the max utility for customer d among $C_2 \cup \{i\}$, i.e., $u_{id} \geq \max_{j \in C_2} u_{jd}$. In this case, $\mathcal{MC}_i^d(C_1) - \mathcal{MC}_i^d(C_2) = (u_{id} - \max_{j \in C_1} u_{jd}) - (u_{id} - \max_{j \in C_2} u_{jd}) \geq 0$;

(ii) location i yields the max utility for customer d among $C_1 \cup \{i\}$ but not $C_2 \cup \{i\}$, i.e., $\max_{j \in C_1} u_{jd} \leq u_{id} < \max_{j \in C_2} u_{jd}$. In this case we have $\mathcal{MC}_i^d(C_1) - \mathcal{MC}_i^d(C_2) = (u_{id} - \max_{j \in C_1} u_{jd}) - \underbrace{(\max_{j \in C_2} u_{jd} - \max_{j \in C_2} u_{jd})}_{=0} \geq 0$;

(iii) location i is not the max utility location for customer d among $C_1 \cup \{i\}$, i.e., $u_{id} < \max_{j \in C_1} u_{jd} \leq \max_{j \in C_2} u_{jd}$. In this case $\mathcal{MC}_i^d(C_1) - \mathcal{MC}_i^d(C_2) = (\max_{j \in C_1} u_{jd} - \max_{j \in C_1} u_{jd}) - (\max_{j \in C_2} u_{jd} - \max_{j \in C_2} u_{jd}) = 0$. In all three cases $\mathcal{MC}_i^d(C_1) - \mathcal{MC}_i^d(C_2) \geq 0$, and hence

Algorithm 4 Efficient Shapley and Banzhaf value Computation for the Facility Location Game

Input: Locations $\mathcal{L}$, customers D, utility matrix U
Output: Shapley and Banzhaf value of all locations

1: Sort the facility locations by (ascending) utility for each customer d, where $U^{d\uparrow}$ is the sorted utility vector of customer d and $\mathcal{L}^{d\uparrow}$ are the sorted facility locations.
2: **for** each location $i \in \mathcal{L}$ **do**
3: $\quad l_i^d \leftarrow$ index of i in $\mathcal{L}^{d\uparrow}$, i.e., $l_i^d = |\mathcal{L}_{id}| - 1$
4: $\quad \varphi_i^{\text{Shapley}} \leftarrow \sum_{d\in D}[\frac{U_{id}}{n-l_i^d+1} - \sum_{t=0}^{l_{id}} \frac{U^{d\uparrow}_{l_i^d-t-1}}{(n-l_i^d+t)+(n-l_i^d+t)^2}]$
5: $\quad \varphi_i^{\text{Banzhaf}} \leftarrow \frac{1}{2^{|\mathcal{L}|-1}} \sum_{d\in D}[2^{|l_i^d|+1}U_{id} - \sum_{t=0}^{l_i^d} U^{d\uparrow}(l_i^d - t + 1)]$
6: **end for**

$\mathcal{MC}_i(C_1) - \mathcal{MC}_i(C_2) = \sum_{d\in D}(\mathcal{MC}_i^d(C_1) - \mathcal{MC}_i^d(C_2)) \geq 0$. And hence we conclude that the facility location game is submodular. □

Next, we compute the Shapley value and Banzhaf value of the players in the facility location game. The following two theorems provide closed-form solutions for the two values, respectively.

Theorem 3.3.2.1. *The Shapley value of a facility location $i \in \mathcal{L}$ can be computed as*

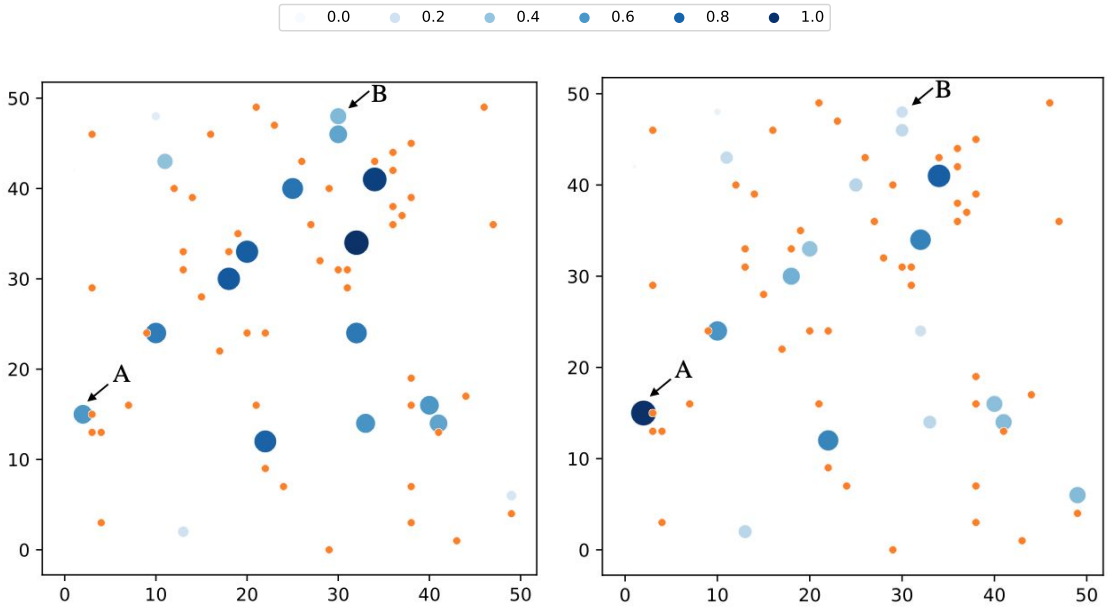
$$\varphi_i^{\text{Shapley}} = \sum_{d\in D}\Big[u_{id}\frac{1}{|\mathcal{L}| - |\mathcal{L}_{id}|} - \sum_{t=1}^{|\mathcal{L}_{id}|}\frac{1}{\lambda(t)+\lambda(t)^2}u_{e_{it}^d}\Big],$$

where $\lambda(t) \coloneqq (|\mathcal{L}| - |\mathcal{L}_{id}| + t - 1)$, $\mathcal{L}_{id} \coloneqq \{j \in \mathcal{L}\setminus\{i\} \mid u_{jd} \leq u_{id}\}$ and $u_{e_{it}^d}$ is the utility value of the t-th largest element after i along the dimension (customer) d.

Proof Sketch. Let $v(C) \coloneqq Fac(C)$ and $n \coloneqq |\mathcal{L}|$ be the number of players. Denote w_C as the weight assigned by the Shapley value to coalition C, i.e., $w_C \coloneqq \frac{1}{n}\binom{n-1}{|C|}^{-1}$. We observe that

$$\varphi_i = \sum_{C\subseteq\mathcal{L}\setminus\{i\}} w_C\mathcal{MC}_i(C) \overset{(*)}{=} \sum_{d\in D}\Big[\underbrace{\sum_{C\subseteq\mathcal{L}_{id}} w_C u_{id}}_{(\#1)} - \underbrace{\sum_{C\subseteq\mathcal{L}_{id}} w_C \max_{j\in C} u_{jd}}_{(\#2)}\Big],$$

where $(*)$ is because the marginal contribution of i to coalition C in dimension d is zero *unless* i is the largest element in C in that dimension and $\mathcal{L}_{id} = \{j \in \mathcal{L} \mid u_{jd} \leq u_{id}\}$ is the coalition of all elements which have values no greater than i in the d-th dimension. Along each dimension d, (#1) is a weighted sum of i's marginal contributions towards coalitions C where i is the largest element; and (#2) sums up for each $j \in \mathcal{L}_{id}$ over all coalitions $C \subseteq \mathcal{L}_{id}$ where j is the largest element. For clarity of the thesis, we provide the detailed subsequent steps in Appendix A.1. □



(a) The Shapley Value **(b)** The Banzhaf value

Figure 3.2: Comparison of the Shapley value and Banzhaf value for the Facility Location Game. The axes correspond to the world axes in a 50x50 map. Orange dots refer to 50 customers, and blue dots refer to 20 facility locations. The size and colour of the facility locations show the respective values, where larger and darker dots have larger values. For a clear comparison, all values are normalised between [0,1].

Theorem 3.3.2.2. *The Banzhaf value of a facility location $i \in \mathcal{L}$ can be computed as*

$$\varphi_i^{\text{Banzhaf}} = \frac{1}{2^{|\mathcal{L}|-1}} \sum_{d \in D} \left[2^{|\mathcal{L}_{id}|} u_{id} - \sum_{t=1}^{|\mathcal{L}_{id}|} 2^{|\mathcal{L}_{id}|-t} u_{e_{it}^d} \right],$$

where $\mathcal{L}_{id} = \{j \in \mathcal{L} \setminus \{i\} \mid u_{jd} \leq u_{id}\}$ and $u_{e_{it}^d}$ is the utility value of the t-th largest element after i for dimension d.

Proof. The proof for the Banzhaf value can be found in Appendix A.1. □

The Shapley value and Banzhaf value of the facility location game can be computed using Theorem 3.3.2.1 and Theorem 3.3.2.2, without needing to enumerate all possible coalitions. Despite the seemingly complex form, the formulas can be implemented straightforwardly by sorting the facility location utility matrix along each dimension (customer). The procedure for computing both the Shapley and Banzhaf values is summarised in Algorithm 4. Compared with a naive approach to computing the Shapley and Banzhaf value by enumerating all possible coalitions, this algorithm significantly improves the computational efficiency.

Figure 3.2 shows a comparison between the Shapley value and the Banzhaf value for the facility location game. Specifically, there are $|\mathcal{L}| = 20$ facility locations (blue dots), and $|D| = 50$

customers (orange dots), all randomly placed in a 50×50 map. We model the utility of a customer d to be decreasing with the Manhattan distance to the facility location i, i.e., $u_{id} = 100 - (|x_i - x_d| + |y_i - y_d|)$. The Shapley value and Banzhaf value are computed using Algorithm 4. In comparison, the Shapley value tend to be higher for locations with large number of nearby customers, while the Banzhaf value puts higher importance on the locations which have many nearby customers but fewer nearby facility locations. For example, location A is distant from nearby facilities, and ranks higher in terms of the Banzhaf value than the Shapley value. In contrast, location B's rank in terms of the Banzhaf value is lower compared to the Shapley value due to the nearby facility. We will further explore the distinction between the Shapley and Banzhaf value in the subsequent sections. Moreover, we will use Algorithm 4 for empirical validations of our results in facility location games with a large number of agents.

3.3.3 Replication Manipulation

Nowadays, in many multiagent ML systems, a player may not only represent a physical entity such as a robot or a human, but also represent digital entities such as an online identity that holds digital resources such as features or data for an ML model. With such digital resources and identities, a player may easily replicate its resource and act under different identities which hold duplicate or overlapping resources.

On the one hand, duplicated resources may happen as a result of malicious manipulations such as the above example where a player creates multiple false identities. On the other hand, duplication may come from an unintentional source and occurs normally in the specific domain. For example, consider an ML prediction task with a set of ML features. There is often redundancy relation between such features. For example, on a medical record, *arterial pressure, arterial bp mean, arterial blood pressure mean* may appear under separate feature entries, but they all share the same semantics [9]. Duplicates can also happen in a more subtle way, such as two features (e.g., birth date and age) that hold similar information. In both the malicious and benign cases, the duplicate resources often do not provide significant additional information for the ML task.

This raises an interesting question: What would happen to the payoff of a player if it holds multiple identities with replicated resources? Can a player increase its payoff by performing malicious replication? To answer these questions, we study the behaviours of the payoff allocation solution

concepts under replication. Specifically, we will model the replication manipulation as a player replicating its resources and acting under multiple identities. This model can be extended in the future to study more complex manipulations or cases in which players have partially overlapping resources. We will also briefly explore some of these possibilities later in Section 3.4.2.

First, we formulate the replication manipulation and define the notion of robustness of the solution concepts under replication. The following definition characterises a replicating player and the induced game as a result of replication.

Definition 3.3.3.1 (Replication Manipulation)**.** *In a submodular game $G = (N, v)$, a (malicious) player i may execute a replication action k times on its resources D_i and acts as $k+1$ players $C^R = \{i_0, i_1, \ldots, i_k\}$ each holding one replica of D_i. Denote the induced game as $G^R = (N^R, v^R)$, where the induced set of players are $N^R = N \setminus \{i\} \cup C^R$, and the induced characteristic function v^R satisfies $\forall C \subseteq N \setminus \{i\}, \forall i_k \in C^R \colon v^R(i_k \cup C) = v(i \cup C)$ and $v^R(C) = v(C)$. By replicating, player i receives a* total payoff *which is the sum of the payoff of all its $k+1$ replicas, i.e., $\varphi_i^{\text{tot}}(k) = \sum_{\kappa=0}^{k} \varphi_{i_\kappa}(N^R, v^R)$.*

In many real-world applications, the payoff allocation component typically has no direct access to the details of replication due to anonymity or the cost of replica detection. Take the online social network for an example, the digital identities of a player is only private and accessible to the player itself, and a single player can create multiple false identities. This fact is captured in the induced game where the replicated identities are treated indifferently by other players. Similarly, the solution concepts will treat the replica identities equally as the other players.

The next assumption captures the fact that adding duplicated resources typically do not (significantly) change the value of the coalition, such as a duplicate feature or replicated data. We refer to this property as *replication redundancy* and formalize it in the following assumption:

Assumption 3.3.3.1 (Replication Redundancy)**.** *A replica does not contribute additional value to coalitions which already contain another replica or the original resource:*

$$\forall i, j \in C^R \colon (i \in C) \rightarrow \mathcal{MC}_j(C) = 0.$$

Despite the fact that having more replicas do not contribute additional value towards the cooperative task, a malicious player may still be able to gain a higher total payoff by replicating,

and the changes in its total payoff vary across different solution concepts. Imagine a malicious player whose objective is to increase its payoff by replication, a payoff allocation method that is robust to the replication can then be used to counteract such malicious behaviours. To this end, we will study the total payoff of the replicating player under various solution concepts. The next definition formalizes the notion of replication-robustness of solution concepts, i.e., a property that ensures that a player through replication gains a total payoff no more than its original payoff.

Definition 3.3.3.2 (Replication-robustness)**.** *A solution concept φ is* replication-robust *if the payoff of the replicating player i in the original game G is no less than the total payoff of the player's replicas C^R in the induced game G^R after replication, i.e.,*

$$\varphi_i(N, v) \geq \sum_{i_\kappa \in C^R} \varphi_{i_\kappa}(N^R, v^R).$$

Having defined the replication manipulation and robustness criteria, we next study the behaviours of different semivalues under replication and their robustness properties.

3.3.4 Payoff Changes for the Shapley Value and Banzhaf Value under Replication, with $k = 1$ Number of Replications

In this section, we will take a look at how the total Shapley value and the Banzhaf value of a player change if it replicates its resource and splits into two identities. To start with, recall that the Shapley value for a player i is defined as

$$\varphi_i^{\text{Shapley}} = \sum_{C \subseteq N \setminus \{i\}} \frac{|C|!(|N| - |C| - 1)!}{|N|!} \mathcal{MC}_i(C)$$

The next lemma presents how much gain the player can obtain by replicating.

Lemma 3.3.4.1. *Let G= (N, v) be a submodular game with replication-redundant characteristic function v, a player $i \in N$ replicates and obtains the total payoff as two identities $C^R = \{i_1, i_2\}$ in the new game $G^R = (N^R, v^R)$, where the players $N^R = N \setminus \{i\} \cup C^R$, and the induced characteristic function v^R satisfies $\forall C \subseteq N \setminus \{i\}$ $\forall i_k \in C^R$: $v^R(i_k \cup C) = v(i \cup C)$ and $v^R(C) = v(C)$. By replicating, the changes in total payoff of player i is:*

$$\delta\varphi_i^{\text{Shapley}} = \sum_{C \subseteq N \setminus \{i\}} \frac{|C|!(|N| - |C| - 1)!}{(|N| + 1)!} (|N| - 2|C| - 1) \mathcal{MC}_i(C)$$

Proof. For the detailed derivation please refer to Appendix A.2 □

Having derived the changes in total payoff, we next show that this value is non-negative for submodular games. This suggests that in a submodular game, under the payoff allocation using Shapley value, a player can gain a higher payoff by replicating.

Lemma 3.3.4.2. *Let G= (N, v) be a submodular game with replication-redundant characteristic function, a player $i \in N$ replicates and obtains the total payoff as two identities $C^R = \{i_1, i_2\}$ in the induced game $G^R = (N^R, v^R)$. Under payoff allocation using the Shapley value, the total payoff of player i after replication is no less than its payoff in the original game.*

Proof. Here we aim to show that the gain of replication is non-negative, i.e.,

$$\delta\varphi_i^{\text{Shapley}} = \sum_{C \subseteq N \setminus \{i\}} \frac{|C|!(|N| - |C| - 1)!}{(|N| + 1)!}(|N| - 2|C| - 1)\mathcal{MC}_i(C) \geq 0$$

To show this inequality, we make use of the submodularity property, which compares the marginal contributions of player i towards pairs of coalitions of other players, i.e., $\forall C_1 \subseteq C_2 \subseteq N \setminus \{i\}, \mathcal{MC}_i(C_1) - \mathcal{MC}_i(C_2) \geq 0$. In order to group coalitions into pairs, we make the following observations on $\delta\varphi_i^{\text{Shapley}}$: Given two coalitions $C_1, C_2 \subseteq N \setminus \{i\}$ with complementary sizes, i.e., $|C_1| + |C_2| = |N \setminus \{i\}| = |N| - 1$,

i. the value of their weights in $\delta\varphi_i^{\text{Shapley}}$ are opposite and adds up to zero, i.e.,

$$\frac{|C_1|!(|N| - |C_1| - 1)!}{|N| + 1!}(|N| - 2|C_1| - 1) = -\frac{|C_2|!(|N| - |C_2| - 1)!}{|N| + 1!}(|N| - 2|C_2| - 1).$$

ii. There are equal number of size c and $|N| - 1 - c$ coalitions because of the symmetry of binomial coefficients $\binom{|N|-1}{c} = \binom{|N|-1}{|N|-1-c}$.

These observations suggest that we may find a one-to-one mapping between each size c coalition and a size $|N| - 1 - c$ superset. Formally, for any coalition size $c < (|N| - 1)/2$, we look for a bijective mapping f between coalitions with inclusion relations and of complementary sizes $f : \{C_1 \subseteq N \setminus \{i\} \mid |C_1| = c\} \mapsto \{C_2 \subseteq N \setminus \{i\} \mid |C_2| = |N| - 1 - c\}$ such that $C_1 \subseteq f(C_1)$. Note that we can omit the corner case where $c = (|N| - 1)/2$ as the weights of these coalitions are equal to zero in $\delta\varphi_i^{\text{Shapley}}$, i.e., $\frac{|C|!(|N|-|C|-1)!}{(|N|+1)!}(|N| - 2|C| - 1) = 0$.

To show the existence of the bijective mapping, we model the coalitions and their inclusion relations ($\subseteq$ and $\supseteq$) by a bipartite graph (Please refer to Figure 3.3 for an illustrated example). For any coalition size $c < (|N| - 1)/2$, define (bipartite) graph $B_c = (L, R, E)$ where each vertex

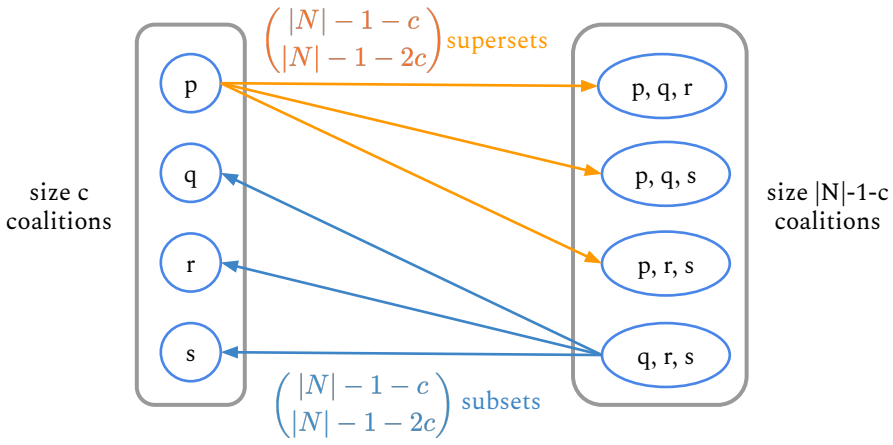


Figure 3.3: Illustration of the proof for Lemma 3.3.4.2. Given an example game with 5 players $N = \{i, p, q, r, s\}$, we match the coalitions (excluding the target player i) of size-c and size-$(|N| - 1 - c)$. (here $c = 1$ and $|N| - 1 - c = 3$). Specifically, each size-c coalition in (L) has $\binom{|N|-c-1}{|N|-2c-1}$ supersets in (R), by adding $|N| - 2c - 1$ members from the remaining players. Conversely, each size-$|N| - c - 1$ coalition in (R) has $\binom{|N|-c-1}{|N|-2c-1}$ subsets of size-c, by choosing and removing $|N| - 2c - 1$ of its members. Arrows indicate the set inclusion relations.

corresponds to a coalition, i.e., vertices $L = \{C_1 \subseteq N \setminus \{i\} \mid |C_1| = c\}$ are the size c coalitions, and vertices $R = \{C_2 \subseteq N \setminus \{i\} \mid |C_2| = |N| - 1 - c\}$ are the size $|N| - 1 - c$ coalitions, and $|L| = |R|$ from observation ii. We then define edges E as the set inclusion relations, that is, $E = \{\{C_1, C_2\} \mid C_1 \in L, C_2 \in R, C_1 \subseteq C_2\}$.

We observe that the graph is k-regular, i.e., every vertex has the same degree. To see that the graph is k-regular, we first show that each coalition $C_1 \in L$ has $\binom{|N|-1-c}{|N|-1-2c}$ supersets in R. Intuitively, to find a size $|N| - 1 - c$ coalition $C_2 \in R$ that is a superset of the size c coalition C_1, we can add $|N| - 1 - 2c$ players by choosing from the remaining $|N| - 1 - c$ players, i.e., $N \setminus \{i\} \setminus \{C_1\}$. Therefore, there are $\binom{|N|-1-c}{|N|-1-2c}$ choices and hence the same number of supersets. Similarly, we can show that each coalition $C_2 \in R$ has $\binom{|N|-1-c}{|N|-1-2c}$ subsets in L. This time, to find a subset of C_2, we can choose and remove $|N| - 1 - 2c$ players from the $|N| - 1 - c$ members of C_2, therefore, there are $\binom{|N|-1-c}{|N|-1-2c}$ choices and hence the same number of subsets. With this we have shown that the bipartite graph is k-regular where $k = \binom{|N|-1-c}{|N|-1-2c}$.

Then, by Hall's Marriage Theorem for regular graphs, there exists a perfect matching on B_c and hence we have found our bijective mapping f. Finally we can pair the terms according to the

mapping. Let $C^c = \{C \subseteq N \setminus \{i\} \mid |C| = c\}$ denote all size c coalitions excluding player i,

$$\begin{aligned}
\delta\varphi_i^{\text{Shapley}} &= \sum_{C \subseteq N \setminus \{i\}} \frac{|C|!(|N| - |C| - 1)!}{(|N| + 1)!}(|N| - 2|C| - 1)\mathcal{MC}_i(C) \\
&= \sum_{0 \leq c < \frac{|N|-1}{2}} \frac{c!(|N| - c - 1)!}{(|N| + 1)!}(|N| - 2c - 1)\left(\sum_{C_1 \in C^c} \mathcal{MC}_i(C_1) - \sum_{C_2 \in C^{|N|-1-c}} \mathcal{MC}_i(C_2)\right) \\
&= \sum_{0 \leq c < \frac{|N|-1}{2}} \frac{c!(|N| - c - 1)!}{(|N| + 1)!}(|N| - 2c - 1) \sum_{C_1 \in C^c} \underbrace{\left(\mathcal{MC}_i(C_1) - \mathcal{MC}_i(f(C_1))\right)}_{\geq 0 \text{ due to submodularity and that } C_1 \subseteq f(C_1)} \\
&\geq 0
\end{aligned}$$

And this concludes our proof that under the Shapley value, the players can gain a higher total payoff by replication. □

Lemma 3.3.4.2 shows that the Shapley value is not robust against replication if the player replicates its resource and acts as two players. This prompts us to ask if there is a solution concept which is robust against replication? The answer is yes, and in what follows we will show that the Banzhaf value is neutral to replication for the case of $k = 1$ number of replications.

Lemma 3.3.4.3. *Let G= (N, v) be a submodular game with replication-redundant characteristic function v, a player $i \in N$ replicates and obtains the total payoff as two identities $C^R = \{i_1, i_2\}$ in the new game $G^R = (N^R, v^R)$, where the players are $N^R = N \setminus \{i\} \cup C^R$, and v^R satisfies $\forall C \subseteq N \setminus \{i\}\ \forall i_k \in C^R$: $v^R(i_k \cup C) = v(i \cup C)$ and $v^R(C) = v(C)$. Under payoff allocation using the Banzhaf value, the changes in total payoff of player i by replicating is zero, i.e., $\delta\varphi_i^{\text{Banzhaf}} = 0$*

Proof. We show that the total payoff of the two replicas is equal to the payoff of the original player. For detailed proof please refer to the Appendix A.2. □

The neutrality of the Banzhaf value under the replication manipulation is a natural consequence of the weights defined on the coalitions (i.e., $\forall C, w_{C,N} = \frac{1}{2^{|N|}}$). This replication neutrality property is also closely related to the 2-efficiency axiom of the Banzhaf value, i.e., the Banzhaf value is neutral to the merging or splitting of two players in the amalgamation game described in Section 2.1.3.

To conclude, the Shapley value is not robust towards replication, while the Banzhaf is neutral when the player replicates and acts as two identities. This raises a few interesting questions: (1) Are there other solution concepts that are robust against replication? (2) What governs the

robustness of the solution concepts which lead to the different behaviours between the Shapley and Banzhaf values? (3) Can we draw the same conclusion for more than one replications, i.e., $k > 1$, for example, is the Banzhaf value neutral to an arbitrary number of replications?

To answer these questions, we next examine the wider class of solution concepts, i.e., semivalues [16, 15], which include both the Shapley and Banzhaf values. Moreover, we extend our theoretical results to the more general case where the player performs $k \geq 1$ replications. In the case of $k > 1$ replications and for general semivalues, the weight changes after replication do not exhibit the convenient form as the Shapley value for matching pairs of coalitions. Therefore, to prove for the general semivalues and $k \geq 1$ replications, we found an alternative solution by representing the semivalues using *average marginal contributions* for the size of each coalition. This representation allows us to transfer the submodularity property which compares pairs of coalitions into a useful inequality on the average marginal contributions which compares pairs of coalition sizes. Based on this inequality, we can then present a general condition for replication-robustness.

3.3.5 Payoff Changes for Semivalues under Replication, with $k \geq 1$ Number of Replications

To compute the payoff (or contribution) of a player towards a cooperative task, we consider *semivalues* [16, 15, 32], a wide class of solution concepts including common ones such as the Shapley value and the Banzhaf value.

3.3.5.1 Semivalues

Given a cooperative game, each semivalue specifies the payoff for a player by a weighted average over its marginal contributions towards coalitions of other players. The weights of player i's marginal contribution towards coalition C is denoted by $w_{C,N}$. In particular, $w_{C,N}$ only depends on the size of C but not on the players' identities inside the coalition, i.e.,

$$\varphi_i(N, v) = \sum_{C \subseteq N \setminus \{i\}} w_{c,N} \mathcal{MC}_i(C), \tag{3.2}$$

where $c = |C|$ is the size of the coalition.

By grouping together equal-sized coalitions, a semivalue assigns to each player a real valued payoff, expressed as a weighted sum of player i's *average marginal contributions towards size-c coalitions* $z_i(c)$:

$$\varphi_i(N,v) = \sum_{c=0}^{|N|-1} \alpha_c z_i(c), \quad \text{where} \tag{3.3}$$

$$z_i(c) = \binom{|N|-1}{c}^{-1} \sum_{C\subseteq N\setminus\{i\},|C|=c} \mathcal{MC}_i(C) \quad \text{(Average Marginal Contributions)},$$

$$\alpha_c = \binom{|N|-1}{c} w_{c,N} \quad \text{(Importance Weights)}.$$

Proof. The derivation from Equation (3.2) to (3.3) is straightforward, by grouping the marginal contributions of player i towards equal-sized coalitions, the payoff of the player φ_i can be re-written as a weighted sum of its average marginal contributions to each coalition size c.

$$\begin{aligned}
\varphi_i(N,v) &= \sum_{C\subseteq N\setminus\{i\}} w_{|C|,N}\mathcal{MC}_i(C) \\
&= \sum_{c=0}^{|N|-1} \sum_{|C|=c, C\subseteq N\setminus\{i\}} w_{|C|,N}\mathcal{MC}_i(C) = \sum_{c=0}^{|N|-1} w_{c,N} \sum_{|C|=c, C\subseteq N\setminus\{i\}} \mathcal{MC}_i(C) \\
&= \sum_{c=0}^{|N|-1} \underbrace{\binom{|N|-1}{c} w_{|C|,N}}_{\alpha_c} \underbrace{\binom{|N|-1}{c}^{-1} \textstyle\sum_{|C|=c, C\subseteq N\setminus\{i\}} \mathcal{MC}_i(C)}_{z_i(c)} = \sum_{c=0}^{|N|-1} \alpha_c z_i(c). \qquad \square
\end{aligned}$$

We will refer to α_c as *importance weights*, as they quantify the importance of a player's marginal contributions towards different coalition sizes. In addition, the importance weights in a semivalue form a probability distribution, that is, $\sum_{c=0}^{|N|-1} \alpha_c = 1$.

The next example compares the importance weights of some common semivalues, namely, the Shapley value, Banzhaf value, and Leave-one-out value.

Example 3.3.5.1 (Importance Weights for Common Semivalues)**.** *The Shapley value is defined by the weights* $w_{c,N} = \frac{c!(|N|-1-c)!}{|N|!} = \frac{1}{|N|}\binom{|N|-1}{c}^{-1}$*, hence the importance weights are* $\alpha_c = \binom{|N|-1}{c} w_{c,N} = \frac{1}{|N|}$ *according to Equation* (3.3)*. Interestingly, the Shapley value has a uniform importance weight, placing equal importance on all coalition sizes. In contrast, the Banzhaf value is defined by the weights* $w_{c,|N|} = \frac{1}{2^{|N|-1}}$*, hence* $\alpha_c = \frac{1}{2^{|N|-1}}\binom{|N|-1}{c}$*. The importance weights of the Banzhaf value is bell-shaped as a function of coalition size, and places higher importance on mid-sized coalitions. For the Leave-one-out value,* $\alpha_c = \mathbb{1}_{c=|N|-1}$.

Intuitively, by adjusting the importance weights α_c, a semivalue balances a player's *individual value* and *complementary value*. In particular, putting higher importance on smaller coalitions (larger α_c for smaller c) favours the individual value and vice-versa. In the next example, we illustrate how to compute the semivalue of a player according to the importance weights and average marginal contributions.

Example 3.3.5.2 (Payoff Computation via Importance Weights)**.** *Let $G = (N, v)$ be a submodular game with 3 players $N = \{i, p, q\}$, the marginal contributions of player i are given by $\mathcal{MC}_i(\emptyset) = 3$, $\mathcal{MC}_i(\{p\}) = \mathcal{MC}_i(\{q\}) = 2$, $\mathcal{MC}_i(\{p, q\}) = 1$. To compute the payoff of player i, we first compute i's average marginal contributions across different coalition sizes $c = 0, 1, 2$, i.e.,*

$$z_i(0) = \mathcal{MC}_i(\emptyset) = 3; z_i(1) = \tfrac{1}{2}(\mathcal{MC}_i(\{p\}) + \mathcal{MC}_i(\{q\})) = 2; z_i(2) = \mathcal{MC}_i(\{p, q\}) = 1.$$

The Shapley value importance weights are uniform, i.e., $\alpha_0 = \alpha_1 = \alpha_2 = \frac{1}{3}$, hence the Shapley value of player i is: $\varphi_i^{\text{Shapley}} = \sum_{c=0}^{2} \alpha_c z_i(c) = \frac{1}{3}(3 + 2 + 1) = 2$. The importance weights of the Banzhaf value are $\alpha_0 = \frac{1}{4}, \alpha_1 = \frac{1}{2}, \alpha_2 = \frac{1}{4}$, and hence $\varphi_i^{\text{Banzhaf}} = \sum_{c=0}^{2} \alpha_c z_i(c) = 3 \cdot \frac{1}{4} + 2 \cdot \frac{1}{2} + \frac{1}{4} = 2$.

So far the representation of semivalues via importance weights has provided some insights for differentiating the common solution concepts. In the following sections, we will show that this representation has significant implications for understanding the behaviours and manipulations of solution concepts in submodular games.

3.3.5.2 Average Marginal Contributions vs. Coalition Sizes

As mentioned previously, in a submodular game, *a player tends to be more useful when contributing towards a smaller coalition.* Can we formally show this intuition? Unfortunately, this does not always hold true for arbitrary pairs of coalitions: provided two different coalitions C_1 and C_2 where $|C_1| \leq |C_2|$, there is no direct comparison between a player's marginal contributions towards these two coalitions, only except for when C_1 is a subset of C_2. Nevertheless, we can formalise this intuition under *average marginal contributions*. We now present in the following a useful property of submodular games that the average marginal contributions $z_i(c)$ decrease with coalition size under the submodularity assumption.

Lemma 3.3.5.1. *Given a submodular game, the average marginal contribution $z_i(c)$ of a player i monotonically decreases with coalition size c, i.e.,*

$$\forall 0 \leq c < |N| - 1, \quad z_i(c) \geq z_i(c + 1). \tag{3.4}$$

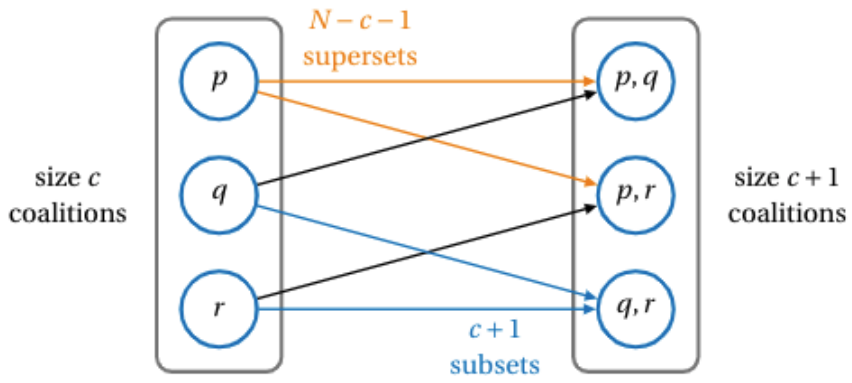


Figure 3.4: Illustration of the Proof for Lemma 3.3.5.1, Step (2): Given an example game with 4 players $N = \{i, p, q, r\}$, we compare the average marginal contribution of player i towards size-c and size-$(c+1)$ coalitions by matching the coalitions. Specifically, each size-c (here $c = 1$) coalition C_1 (Left) has $|N| - c - 1$ supersets of size-$(c+1)$. This can be shown by adding any of the remaining $|N| - c - 1$ players ($-c$ refers to the c players already in the coalition and -1 refers to the player i). Conversely, each size-$(c+1)$ coalition C_2 (Right) has $c+1$ subsets of size-c. This can be shown by removing any one of its $c+1$ members. Arrows indicate the "$\subseteq$" relation.

Proof. (Sketch) To show for any player i, $z_i(c) \geq z_i(c+1)$, we take the following three steps:

(1) Express $z_i(c)$ and $z_i(c+1)$ as the average of marginal contributions of player i towards size-c and size-$(c+1)$ coalitions, respectively.

(2) Map the size-c coalitions (excluding i) with their size-$(c+1)$ supersets (excluding i), and vice-versa: each size-c coalition C_1 can be mapped to $(|N| - 1 - c)$ number of size-$(c+1)$ supersets C_2 where $C_1 \subseteq C_2 \subseteq N \setminus \{i\}$. This can be achieved by adding one of the remaining $(|N| - 1 - c)$ elements $j \in N \setminus (\{i\} \cup C_1)$. Conversely, each C_2 can be mapped to $(c+1)$ subsets C_1 of size-c. This can be achieved by removing any one of the member elements $j \in C_2$. An example is shown in Figure 3.4 for an illustration. Note that in contrast to the proof for Lemma 3.3.4.2, here we are not looking for a bijective mapping.

(3) Finally, with the mappings between size c and $c+1$ coalitions, we show that $z_i(c) \geq z_i(c+1)$ by the submodularity property: $\forall C_1 \subseteq C_2 \subseteq N \setminus \{i\} \implies \mathcal{MC}_i(C_1) \geq \mathcal{MC}_i(C_2)$.

The detailed derivations of Step (3) is shown in the following: $\forall c \in [0, 1, \ldots, |N| - 2]$, denote $C^c := \{C \subseteq N \setminus \{i\} \mid |C| = c\}$ as all possible coalitions of size c, excluding player i, then

$$
\begin{aligned}
z_i(c+1) - z_i(c) &= \sum_{C_2 \in C^{c+1}} \binom{|N|-1}{c+1}^{-1} \mathcal{MC}_i(C_2) - \sum_{C_1 \in C^c} \binom{|N|-1}{c}^{-1} \mathcal{MC}_i(C_1) \\
&= \sum_{C_2 \in C^{c+1}} \Bigg(\binom{|N|-1}{c+1}^{-1} \mathcal{MC}_i(C_2) - \sum_{C_1 \in C^c, C_1 \subseteq C_2} \underbrace{\tfrac{1}{|N|-1-c}}_{(1)} \binom{|N|-1}{c}^{-1} \underbrace{\mathcal{MC}_i(C_1)}_{\geq \mathcal{MC}_i(C_2)} \Bigg) \\
&\leq \sum_{C_2 \in C^{c+1}} \Bigg(\binom{|N|-1}{c+1}^{-1} \mathcal{MC}_i(C_2) - \sum_{C_1 \in C^c, C_1 \subseteq C_2} \tfrac{1}{|N|-1-c} \binom{|N|-1}{c}^{-1} \mathcal{MC}_i(C_2) \Bigg) \\
&= \sum_{C_2 \in C^{c+1}} \Bigg(\binom{|N|-1}{c+1}^{-1} \mathcal{MC}_i(C_2) - \underbrace{\tfrac{c+1}{|N|-1-c}}_{(2)} \binom{|N|-1}{c}^{-1} \mathcal{MC}_i(C_2) \Bigg) \\
&= \sum_{C_2 \in C^{c+1}} \Bigg(\binom{|N|-1}{c+1}^{-1} \mathcal{MC}_i(C_2) - \binom{|N|-1}{c+1}^{-1} \mathcal{MC}_i(C_2) \Bigg) \\
&= 0
\end{aligned}
$$

(1) C_1 is counted once in each of its $(|N| - 1 - c)$ supersets C_2 of size-$(c + 1)$, and (2) is because each C_2 has $c + 1$ subsets C_1 of size-c. And this concludes our proof for $z_i(c) \leq z_i(c + 1)$. □

In summary, we have shown that in a submodular game, a player is more useful *on average* when contributing towards a smaller coalition, i.e., the player's average marginal contribution towards a smaller coalition is no less than its average marginal contribution to a bigger coalition.

3.3.5.3 Payoff Changes under Replication with $k \geq 1$

In Section 3.3.4, we have shown that the Shapley is not robust against replication when the player replicates once and acts as two new players. Now we are ready to extend our study to the more general class of semivalues and number of replications $k \geq 1$. As we discussed previously, it is important for the solution concept to hold robust against any number of replications k, as the payoff allocation module may not have access to the private information of how many replications the player has performed and which identities are the replicas.

As a first step, we derive the total payoff of the replicating player according to each solution concept after replication. Interestingly, we observe that under the replication-redundancy assumption, the replicating player's total payoff can be expressed by the average marginal contributions $z_i(c)$ of the original player *from the original game*, as detailed in the following lemma.

Lemma 3.3.5.2. *Let $G = (N, v)$ be a submodular game with replication-redundant characteristic function v. By replicating k times and acting as $k + 1$ players $C^R = \{i_0, \ldots, i_k\}$ in the induced*

game $G^R = (N^R, v^R)$, the malicious player i receives a total payoff of

$$\varphi_i^{\text{tot}}(k) = \sum_{c=0}^{|N|-1} \alpha_c^k z_i(c), \text{ where} \tag{3.5}$$

$$z_i(c) = \binom{|N|-1}{c}^{-1} \sum_{C \subseteq N \setminus \{i\}, |C|=c} \mathcal{MC}_i(C) \quad \text{(avg. marginal contributions in } G\text{)},$$

$$\alpha_c^k = (k+1)\binom{|N|-1}{c} w_{c,N^R} \quad \text{(importance weights after replication)}.$$

Proof Sketch. By symmetry the replicas yield equal payoff, i.e., $\varphi_i^{\text{tot}}(k) = (k+1)\varphi_{i_k}(N^R, v^R)$. Due to replication-redundancy (Assumption 3.3.3.1), a replica player makes a nonzero marginal contribution only towards coalitions with no other replicas $C \subseteq N^R \setminus C^R$, which correspond to the same set of coalitions of the other players in the original game $C \subseteq N \setminus \{i\}$ because $N^R \setminus C^R = N \setminus \{i\}$. Then we can compute the new importance weights α_c^k over the original coalitions. The detailed proof can be found in Appendix A.3. □

Note that Equation (3.5) reduces to Equation (3.3) for no replications, i.e., $\alpha_c^k = \alpha_c$ when $k = 0$. Importantly, $z_i(c)$ are the average marginal contributions defined on the original game $G = (N, v)$ as in Equation (3.3), instead of on the induced game, thus they *are invariant under replication*. As stated in Equation (3.5), the total payoff of the replicating player is a weighted sum over $z_i(c)$ through the new importance weights α_c^k. Since the average marginal contributions $z_i(c)$ in the original game stay invariant after replication, the change in the total payoff of the replicating player φ_i^{tot} is reflected in the change in α_c^k across different number of replications k. This makes α_c^k a key factor for characterising replication-robustness.

The next corollary demonstrates the importance weights after replication for the common semivalues.

Corollary 3.3.5.1. *The total payoff of a malicious player i after k replications can be expressed as $\varphi_i^{\text{tot}}(k) = \sum_{c=0}^{N-1} \alpha_c^k z_i(c)$ such that for the Leave-one-out value $\alpha_c^k = \mathbb{1}_{c=|N|-1, k=0}$, for the Shapley value $\alpha_c^k = \frac{(k+1)\binom{|N|-1}{c}}{(|N|+k)\binom{|N|+k-1}{c}}$, and for the Banzhaf value $\alpha_c^k = \frac{(k+1)}{2^{|N|+k-1}}\binom{|N|-1}{c}$.*

Proof. The importance weights can be obtained straightforwardly by plugging in the weights of the solution concepts to Equation (3.5). The detailed proof is omitted, but an illustration can be found in Example 3.3.5.3. □

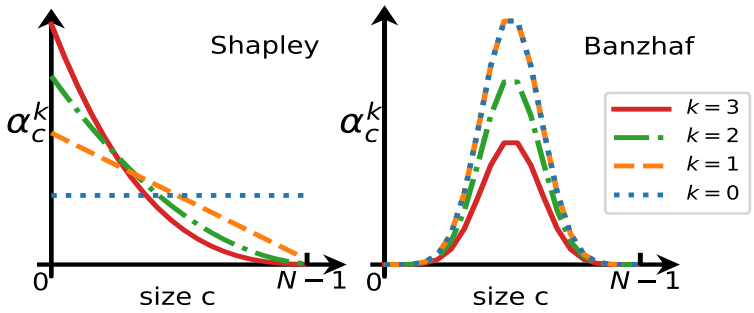


Figure 3.5: Changes of α_c^k under different number of replications k, plot using Equation (3.5) with $|N| = 20$. The x-axis represents the sizes c of coalitions of the other players $N \setminus \{i\}$, and the y-axis shows the value of the importance weights α_c^k assigned to each coalition size. Each curve represents a different number of replications k. (Left) Across the different curves, the importance weights α_c^k of Shapley shift towards smaller coalitions as k increases (Lemma 3.3.6.1). (Right) In contrast, the Banzhaf importance weights α_c^k are unchanged with the first replication, afterwards, α_c^k decreases across all coalition sizes as k increases. Since $z_i(c)$ decreases over coalition size c due to the submodular characteristic function (Lemma 3.3.5.1), the weight shift of Shapley value causes φ_i^{tot} to be increasing, while non-increasing for the Banzhaf value.

In Example 3.3.5.3 and Figure 3.5, we compare the Shapley value and the Banzhaf value using Equation 3.5, and illustrate the difference between these two solution concepts in terms of their changes in importance weights.

Example 3.3.5.3 (Payoff Changes of the Malicious Player)**.** *Let $G = (N, v)$ be a submodular game with 3 players $N = \{i, p, q\}$. The marginal contributions of player i towards coalitions of other players are $\mathcal{MC}_i(\emptyset) = 3$, $\mathcal{MC}_i(\{p\}) = \mathcal{MC}_i(\{q\}) = 2$, $\mathcal{MC}_i(\{p, q\}) = 1$. Player i replicates once and acts under two identities $C^R = \{i_1, i_2\}$. The induced game is then $G^R = (N^R, v^R)$ where $N^R = \{i_1, i_2, p, q\}$. To see the changes in i's total payoff, we take the following steps:*

(1) First, compute the average marginal contributions of i in the original game:

$$z_i(0) = \mathcal{MC}_i(\emptyset) = 3; \quad z_i(1) = \tfrac{1}{2}(\mathcal{MC}_i(p) + \mathcal{MC}_i(q)) = 2; \quad z_i(2) = \mathcal{MC}_i(p, q) = 1.$$

(2) Then, compute the importance weights in the induced game according to Equation (3.5)*:*

$$\textit{(Shapley)}\ w_{c,N^R} := \tfrac{1}{|N|+k}\binom{|N|+k-1}{c}^{-1} \implies \alpha_c^k = \tfrac{k+1}{|N|+k}\binom{|N|-1}{c}\binom{|N|+k-1}{c}^{-1}$$

$$\textit{(Banzhaf)}\ w_{c,N^R} := \tfrac{1}{2^{|N|+k-1}} \implies \alpha_c^k = \tfrac{k+1}{2^{|N|+k-1}}\binom{|N|-1}{c}.$$

(3) Finally, we compute the total payoffs of player i using $\varphi_i^{\text{tot}}(k) = \sum_{c=0}^{|N|-1} \alpha_c^k z_i(c)$, where $k = 0$ refers to no replication, and $k > 0$ represents replicating k times:

$$\textit{(Shapley)}\ \varphi_i^{\text{tot}}(0) = \textstyle\sum_{c=0}^{2} \alpha_c^0 z_i(c) = 3 \cdot \frac{1}{3} + 2 \cdot \frac{1}{3} + 1 \cdot \frac{1}{3} = 2;$$

$$\varphi_i^{\text{tot}}(1) = \textstyle\sum_{c=0}^{2} \alpha_c^1 z_i(c) = 3 \cdot \frac{1}{2} + 2 \cdot \frac{1}{3} + 1 \cdot \frac{1}{6} = \frac{7}{3}.$$

$$\textit{(Banzhaf)}\ \varphi_i^{\text{tot}}(0) = \textstyle\sum_{c=0}^{2} \alpha_c^0 z_i(c) = 3 \cdot \frac{1}{4} + 2 \cdot \frac{1}{2} + 1 \cdot \frac{1}{4} = 2;$$

$$\varphi_i^{\text{tot}}(1) = \sum_{c=0}^{2} \alpha_c^1 z_i(c) = 3 \cdot \tfrac{1}{4} + 2 \cdot \tfrac{1}{2} + 1 \cdot \tfrac{1}{4} = 2.$$

The above example demonstrates our observations from Figure 3.5: the importance weights α_c^k of Shapley value shifts from uniform in the coalition sizes $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$, to having larger weights towards smaller coalition sizes $(\frac{1}{2}, \frac{1}{3}, \frac{1}{6})$ after replication. Due to submodularity, the average marginal contributions are larger for smaller coalition sizes, hence the total payoff increases as a result of replication. Whereas for the Banzhaf value, the importance weights $\alpha_c^k = (\frac{1}{4}, \frac{1}{2}, \frac{1}{4})$ are invariant under the first replication, hence the total payoff is unchanged. The following section will provide a formal characterisation of these observations.

3.3.6 Replication-robustness Condition

In this section, we will present the condition which characterises the replication-robustness for semivalues. Specifically, this condition provides a sufficient and necessary condition on the importance weights α_c^k for guaranteeing replication-robustness against any arbitrary number of replications k. Therefore, by using a replication-robust solution concept, a player should have no incentive to perform any number of replications in order to increase its payoff.

3.3.6.1 Necessary and Sufficient Condition

Theorem 3.3.6.1. *Given a submodular game with replication-redundant characteristic function, a semivalue of the form* $\varphi_i = \sum_{c=0}^{|N|-1} \alpha_c z_i(c)$ *is replication-robust* if and only if *for any number of replications k,*

$$\forall 0 \leq p \leq |N| - 1, \sum_{c=0}^{p} \alpha_c^0 \geq \sum_{c=0}^{p} \alpha_c^k, \tag{3.6}$$

where α_c^k *are the importance weights as defined in Equation* (3.5).

Proof. **Sufficiency.** We will first show the sufficient condition, that is, Equation (3.6) implies replication-robustness, i.e.,:

$$\forall k, (\forall 0 \leq p \leq N - 1, \sum_{c=0}^{p} \alpha_c^0 \geq \sum_{c=0}^{p} \alpha_c^k) \implies (\varphi_i^{\text{tot}}(0) - \varphi_i^{\text{tot}}(k) \geq 0)$$

Due to submodularity, the average marginal contributions decrease as coalition sizes grow according to Lemma 3.3.5.1, together with replication-redundancy, we have the average marginal contributions satisfy the following condition $z_i(0) \geq \ldots \geq z_i(|N| - 1) \geq 0$.

Let $\delta_c = \alpha_c^0 - \alpha_c^k$ denote the change in importance weight over coalition size c before and after replication, then by Equation (3.6), $\forall p \in \{0, 1, \ldots, |N| - 1\}, \sum_{c=0}^{p} \delta_c \geq 0$. Therefore, we proceed to show the inequality recursively as follows:

$$
\begin{aligned}
\varphi_i^{\text{tot}}(0) - \varphi_i^{\text{tot}}(k) &= \sum_{c=0}^{|N|-1} \delta_c z_i(c) \\
&= z_i(0) \sum_{c=0}^{0} \delta_c + \sum_{c=1}^{|N|-1} \delta_c z_i(c) && \text{, by grouping coalitions of size } c = 0 \text{ and } c > 0 \\
&\geq z_i(1) \sum_{c=0}^{0} \delta_c + \sum_{c=1}^{|N|-1} \delta_c z_i(c) && \text{, because } z_i(0) \geq z_i(1) \text{ and } \sum_{c=0}^{0} \delta_c \geq 0, \\
&= z_i(1) \sum_{c=0}^{1} \delta_c + \sum_{c=2}^{|N|-1} \delta_c z_i(c) && \text{, moving size } c+1 \text{ to } \{0, \ldots, c\} \\
&\geq z_i(2) \sum_{c=0}^{1} \delta_c + \sum_{c=2}^{|N|-1} \delta_c z_i(c) \geq \ldots && \text{, because } z_i(1) \geq z_i(2) \text{ and } \sum_{c=0}^{1} \delta_c \geq 0 \\
&= z_i(|N|-2) \sum_{c=0}^{|N|-2} \delta_c + \sum_{c=|N|-1}^{|N|-1} \delta_c z_i(|N|-1) \\
&\geq z_i(|N|-1) \sum_{c=0}^{|N|-2} \delta_c + \sum_{c=|N|-1}^{|N|-1} \delta_c z_i(|N|-1) \\
&= z_i(|N|-1) \sum_{c=0}^{|N|-1} \delta_c \\
&\geq 0.
\end{aligned}
$$

With this, we have shown the sufficient condition, and we will next show the necessary condition.

Necessity. We now show that Equation (3.6) is also a necessary condition. To do this, we will prove by contradiction:

Let $\tilde{\delta}_c^k := \tilde{\alpha_c^0} - \tilde{\alpha_c^k}$. Recall in Equation (3.6) for any coalition size $0 \leq p \leq |N|-1, \sum_{c=0}^{q} \tilde{\delta}_c^k \geq 0$. Assume the contrary that there exists a set of coalition sizes (index) q_m where the condition does not hold:

$$
\exists Q_m = \{q_0, q_1, \ldots, q_m\}, \text{ such that } \forall q \in Q_m, \sum_{c=0}^{q} \tilde{\delta}_c^k < 0,
$$

Without loss of generality, we assume the coalition sizes are ordered and that $q_0 < q_1 < \ldots < q_m \leq |N| - 1$. We now show that there exist average marginal contributions $z_i(c)$'s which violates replication-robustness, and we construct them as follows: Looking at the smallest index that causes the contrary assumption $q_0 = \min Q_m$, all indices below q_0 satisfy the original condition,

i.e.,

$$\sum_{c=0}^{p} \tilde{\delta}_c^k \begin{cases} < 0 & if\, p = q_0 \\ \geq 0 & if\, p < q_0 \end{cases}$$

Therefore, the sum of $\tilde{\delta_c^k}$ over all indices under q_0 is less than the absolute value of that at q_0, i.e., $0 \leq \sum_{c=0}^{q_0-1} \tilde{\delta}_c^k < -\tilde{\delta}_{q_0}^k = |\tilde{\delta}_{q_0}^k|$. Therefore, we denote $\gamma < 1$ as the ratio, such that $\sum_{c=0}^{q_0-1} \tilde{\delta}_c^k = \gamma|\tilde{\delta}_{q_0}^k|$. To construct $z_i(c)$, we let $\forall c > q_0, z_i(c) = 0$ for all indices above q_0, and let $z_i(q_0) = \gamma z_i(0) + \epsilon$ where $0 < \epsilon \leq (1-\gamma)z_i(0)$. Note that the $\epsilon > 0$ is for $z_i(q_0)$ to be strictly greater than $z_i(0)$, and $\epsilon \leq (1-\gamma)z_i(0)$ guarantees submodularity where $z_i(q_0) \leq z_i(0)$. In fact, a trivial choice would be a constant function for all indices no greater than q_0, i.e., $\forall q \in \{0, \ldots, q_0\}, z_i(q) = Const.$, but we will adopt the former option which also accounts for strictly submodular cases. Then, we have

$$\begin{aligned}
\tilde{\varphi}_i^{\text{tot}}(0) - \tilde{\varphi}_i^{\text{tot}}(k) &= \sum_{c=0}^{|N|-1} \tilde{\delta}_c^k z_i(c) \\
&= \sum_{c=0}^{q_0} \tilde{\delta}_c^k z_i(c) && \text{, as } \forall q_0 < q \leq |N|-1, z_i(q) = 0 \\
&= (\sum_{c=0}^{q_0-1} \tilde{\delta}_c^k z_i(c)) + \tilde{\delta}_{q_0}^k z_i(q_0) \\
&\leq (\sum_{c=0}^{q_0-1} \tilde{\delta}_c^k) z_i(0) + \tilde{\delta}_{q_0}^k z_i(q_0) && \text{, due to submodularity} \\
&= |\tilde{\delta}_{q_0}^k|(\gamma z_i(0) - z_i(q_0)) \\
&= |\tilde{\delta}_{q_0}^k|(\gamma z_i(0) - \gamma z_i(0) - \epsilon) \\
&= -\epsilon < 0
\end{aligned}$$

This forms a contradiction. Thus we have proven that Theorem 3.3.6.1 is both a necessary and sufficient condition for replication-robustness, and this concludes our proof. □

Intuitively, the condition ensures that after replication, there would not be too much importance weight increase on the smaller coalition sizes. This is because players have larger average marginal contributions $z_i(c)$ towards smaller coalitions as a result of submodularity. This effect was illustrated in Figure 3.5 and the theorem is a formal characterisation.

The significance of this theorem is that it provides a necessary and sufficient condition for guaranteeing replication-robustness for all semivalues and for any number of replications. Therefore,

a solution concept that satisfies the condition is replication-robust without the need of knowing the number of replications k or the replicated identities, which are typically private to the player.

3.3.6.2 Sufficient Conditions for Monotonically Increasing (Decreasing) Total Payoff

Extending from the necessary and sufficient condition in Theorem 3.3.6.1, the following two corollaries provide sufficient conditions for monotonically increasing or monotonically decreasing total payoff of the replicating player with respect to the number of replications. These conditions will help us characterise the robustness of the common semivalues.

1. Monotonica Decreasillyng Total Payoff vs. the Number of Replications k

Corollary 3.3.6.1. *Given a submodular game with a replication-redundant characteristic function, a semivalue is replication-robust and the total payoff of the malicious player decreases monotonically i.e.,*$\varphi_i^{\text{tot}}(k) \geq \varphi_i^{\text{tot}}(k+1)$, if *for any number of replications k,*

$$\forall 0 \leq p \leq N-1, \sum_{c=0}^{p} \alpha_c^k \geq \sum_{c=0}^{p} \alpha_c^{k+1} \tag{3.7}$$

Note that the condition stated in Corollary 3.3.6.1 is stricter than that in Theorem 3.3.6.1 (which implied $\varphi_i^{\text{tot}}(0) \geq \varphi_i^{\text{tot}}(k)$), but additionally ensures that the total payoff of a replicating player monotonically decreases with the number of replications, i.e., $\forall k \geq 0, \varphi_i^{\text{tot}}(k) \geq \varphi_i^{\text{tot}}(k+1)$.

Proof Sketch. We need to show that Equation (3.7) implies the total payoff decreases with k:

$$\forall k, \varphi_i^{\text{tot}}(k) - \varphi_i^{\text{tot}}(k+1) = \sum_{c=0}^{|N|-1} (\alpha_c^k - \alpha_c^{k+1}) z_i(c) \geq 0,$$

Denote $\delta_c^k := \alpha_c^k - \alpha_c^{k+1}$, then we can substitute δ_c^k in the recursive proof for the sufficient condition of Theorem 3.3.6.1, by doing so we will arrive at the above conclusion. □

2. Monotonically Increasing Total Payoff vs. the Number of Replications k

Corollary 3.3.6.2. *Given a submodular game with a replication-redundant characteristic function, a semivalue is not replication-robust, and the total payoff of the malicious player increases monotonically i.e.,*$\varphi_i^{\text{tot}}(k) \leq \varphi_i^{\text{tot}}(k+1)$, if *for any number of replications k,*

$$\forall 0 \leq p \leq N-1, \sum_{c=0}^{p} \alpha_c^{k+1} \geq \sum_{c=0}^{p} \alpha_c^k \tag{3.8}$$

Proof Sketch. We need to show that Equation (3.8) implies the total payoff increases with k:

$$\forall k, \varphi_i^{\text{tot}}(k+1) - \varphi_i^{\text{tot}}(k) = \sum_{c=0}^{|N|-1} (\alpha_c^{k+1} - \alpha_c^k) z_i(c) \geq 0,$$

The proof is similar to the monotonically increasing case above. Denote $\delta_c^k \coloneqq \alpha_c^{k+1} - \alpha_c^k$, and we can reuse the proof for Theorem 3.3.6.1. □

3.3.6.3 Robustness of Common Solution Concepts

In Section 3.3.4 we discussed the payoff changes using the Shapley value under $k = 1$ replication. Now using the robustness conditions presented in the previous section, we can revisit the Shapley value and the Banzhaf value with $k \geq 1$ replications. The following theorem is a consequence of our robustness condition for the three common semivalues.

Theorem 3.3.6.2. *Let $G = (N, v)$ be a submodular game where v is replication-redundant, the Shapley value is not replication-robust, whereas the Banzhaf value and Leave-one-out are replication-robust. For the Shapley value, the total payoff of the replicating player i monotonically increases over the number of replicas, and converges to i's characteristic value, i.e.,* $\lim_{k\to\infty} \varphi_i^{\text{tot}}(k) = v(\{i\})$. *For the Banzhaf and Leave-one-out values,* $\lim_{k\to\infty} \varphi_i^{\text{tot}}(k) = 0$.

Proof. **For the Shapley Value.** We prove that the Shapley value is not replication-robust, and the total payoff of the replicating player monotonically increases with k in the following three steps:

1. Rewrite the Shapley value after replication according to Lemma 3.3.5.2 in terms of average marginal contributions and importance weights, i.e., $\alpha_c^k = \frac{(k+1)}{(|N|+k)} \binom{|N|-1}{c} \binom{|N|+k-1}{c}^{-1}$.

2. In Lemma 3.3.6.1 (presented following this theorem), we show that the Shapley value satisfies Equation (3.9b):

$$\forall 0 \leq p \leq N-1, \quad \sum_{c=0}^{p} \alpha_c^k \leq \sum_{c=0}^{p} \alpha_c^{k+1}.$$

3. By Theorem 3.3.6.1, the Shapley value is not replication-robust. In addition, Corollary 3.3.6.1 in Section 3.3.6.2 shows that for the Shapley value, the total payoff of the replicating player monotonically increases with respect to increasing number of replications k.

Finally, the limit is computed as follows:

$$\begin{aligned}\lim_{k\to\infty}\varphi_i^{\text{tot}}(k) &= \lim_{k\to\infty}\sum_{c=0}^{|N|-1}\tfrac{k+1}{|N|+k}\binom{|N|-1}{c}\binom{|N|+k-1}{c}^{-1}z_i(c)\\ &= \sum_{c=0}^{|N|-1}\binom{|N|-1}{c}z_i(c)\underbrace{\lim_{k\to\infty}\tfrac{k+1}{|N|+k}}_{=1}\underbrace{\lim_{k\to\infty}\binom{|N|+k-1}{c}^{-1}}_{=\mathbb{1}_{c=0}}\\ &= z_i(0) = \mathcal{MC}_i(\emptyset) = v(\{i\})\end{aligned}$$

(ii) For the Banzhaf value and Leave-one-out value. We prove that the importance weights α_c^k satisfy $\forall k \geq 0, \alpha_c^k \geq \alpha_c^{k+1}$ for both the Banzhaf value and Leave-one-out, which is a sufficient condition for $\forall p, \sum_{c=0}^{p}\alpha_c^0 \geq \sum_{c=0}^{p}\alpha_c^k$. Therefore, according to our robustness condition in Theorem 3.3.6.1, both the Banzhaf value and Leave-one-out value are replication-robust. In addition, both values monotonically decrease as the number of replications k according to Corollary 3.3.6.1. In particular, for the Banzhaf value,

$$\begin{cases}\alpha_c^k = \frac{(k+1)}{2^{|N|+k-1}}\binom{|N|-1}{c}\\ \alpha_c^{k+1} = \frac{(k+2)}{2^{|N|+k}}\binom{|N|-1}{c}\end{cases} \quad\Longrightarrow\quad \frac{\alpha_c^k}{\alpha_c^{k+1}} = 2\tfrac{(k+1)}{(k+2)} \geq 1$$

As we can see, $\alpha_c^0 = \alpha_c^1$ for the Banzhaf value. This implies that the total payoff of the malicious player is unchanged when it replicates for the first time. Therefore, the Banzhaf value is neutral for $k = 1$, which conforms with our finding in Lemma 3.3.4.3. The limit of the total payoff is:

$$\lim_{k\to\infty}\varphi_i^{\text{tot}}(k) = \lim_{k\to\infty}\sum_{c=0}^{|N|-1}\alpha_c^k z_i(c) = \sum_{c=0}^{|N|-1}\binom{|N|-1}{c}z_i(c)\lim_{k\to\infty}\frac{k+1}{2^{|N|+k-1}} = 0$$

For the Leave-one-out value. $\forall k \geq 0, \frac{\alpha_c^{k+1}}{\alpha_c^k} = 0$, hence $\frac{\alpha_c^k}{\alpha_c^{k+1}} \geq 1$, and the total payoff is zero with any positive number of replications, i.e., $\forall k > 0, \varphi_i^{\text{tot}}(k) = 0$ □

As we mentioned previously in Section 3.3.5, the semivalues balance between a player's *individual value* and *complementary value* through the importance weights α_c. The Shapley value assigns uniform weights over coalition sizes c, Banzhaf value assigns larger weights for mid-sized coalitions, and Leave-one-out only considers size $N-1$ coalitions. We observe that due to replication-redundancy and submodularity, the solution concepts which emphasize the complementary value tend to be more replication-robust.

In particular, the robustness property of the Shapley value aligns with and generalises our findings for the $k = 1$ case. With an increasing number of replications, the player's total payoff

monotonically increases and converges to its own characteristic value. This is due to the following properties shown in the next lemma.

Lemma 3.3.6.1. *For the Shapley value, the importance weights α_c^k of the total payoff of a replicating player satisfy the following properties:* $\forall k \geq 0, \forall 0 \leq p \leq |N| - 1$,

$$\sum_{c=0}^{|N|-1} \alpha_c^k = 1 \tag{3.9a}$$

$$\sum_{c=0}^{p} \alpha_c^k \leq \sum_{c=0}^{p} \alpha_c^{k+1} \tag{3.9b}$$

$$\sum_{c=0}^{p} \alpha_c^{k+1} - \alpha_c^k \geq \sum_{c=0}^{p} \alpha_c^{k+2} - \alpha_c^{k+1} \tag{3.9c}$$

Proof. For the detailed proof please refer to Appendix A.4. □

Equation (3.9a) presents an interesting phenomenon that the importance weights of the Shapley value after replication (i.e., α_c^k) always sum to 1. This is a special property of the Shapley value which is not shared by all semivalues. Please note the difference from the property of semivalues, i.e., $\sum_{c=0}^{|N|-1} \alpha_c = 1$. In contrast, Equation (3.9a) shows the sum of the importance weights after replication α_c^k over the coalitions in the original game $C \subseteq N \setminus \{i\}$. Equation (3.9b) shows that these importance weights gradually *shifts* towards the smaller coalitions with each added replication, which results in the monotonically increasing total payoff. Collectively, Equation (3.9a) and Equation (3.9b) results in the convergence of the malicious player's total payoff to its characteristic value. Additionally, Equation (3.9c) implies that the gain of adding one replica decreases with increasing number of replications, hence the first replication yields the malicious player the highest unit gain.

To summarise, in this section, we have studied and compared the replication-robustness properties of the common semivalues, namely, the Shapley value, the Banzhaf value and the Leave-one-out value. In the following section, we discuss the design of other replication-robust solution concepts.

3.3.6.4 Designing Customized Replication-robust Payoff Allocations

In this section, we describe how to apply the replication-robustness condition to the design of new robust solution concepts and illustrate this with an example robust solution concept derived from the Shapley value.

Observation 1: To satisfy the robustness conditions in Theorem 3.3.6.1, it suffices to satisfy one of the following conditions for each summand of coalition size c:

$$\alpha_c^0 \geq \alpha_c^k, \text{ or monotonicity: } \alpha_c^k \geq \alpha_c^{k+1} \tag{3.10}$$

Observation 2: The identity of the replicating player and the number of replicas k are private information that is often not accessible. Therefore, we should make sure that k does not appear in the solution concept.

We now derive a robust solution by down-weighing the Shapley value using these two observations. Our solution will take the following form, where the factor $\gamma_{|N|}^{|C|}$ is a function of the total number of players $|N|$ and coalition size $|C|$, but not of the number of replicas k:

$$\tilde{\varphi}_i(N, v) \coloneqq \sum_{C \subseteq N \setminus \{i\}} \gamma_{|N|}^{|C|} w_{|C|,N} \mathcal{MC}_i(C) \tag{3.11}$$

where $w_{|C|,N} = \frac{|C|!(|N|-|C|-1)!}{|N|!}$ are the Shapley coefficients.

Definition 3.3.6.1. *(Robust Shapley value) Equation (3.11) with*

$$\gamma_{|N|}^{|C|} = \begin{cases} \frac{\lceil \frac{|N|-1}{2} \rceil ! \lfloor \frac{|N|-1}{2} \rfloor !}{|C|!(|N|-|C|-1)!} & \text{if } |C| < \lfloor \frac{|N|-1}{2} \rfloor, \\ 1 & \text{otherwise.} \end{cases}$$

defines the Robust Shapley value.

Corollary 3.3.6.3. *The Robust Shapley value in Definition 3.3.6.1 is replication-robust. Moreover, in a submodular game $G = (N, v)$, the loss for a replicating player i by replicating k times is at least:* $\varphi_i^{\text{tot}}(0) - \varphi_i^{\text{tot}}(k) \geq \frac{1}{|N|} \sum_{c=0}^{|N|-1} (1 - \frac{k+1}{2^k}) \gamma_{|N|}^c z_i(c)$.

Proof. For the proof of this corollary please refer to Appendix A.5 □

The Robust Shapley value satisfies axioms symmetry **(A1)**, null-player **(A3)**, linearity **(A4)**. And additionally, the total allocated payoff does not exceed the value of the grand coalition $v(N)$.

Like the Banzhaf value and Robust Shapley value, there are many other possible solution concepts which are replication-robust. These solution concepts can be crafted by the importance weights. Recall that semivalues balance between a player's individual value and complementary value through the importance weights. As a rule of thumb, the solution concepts which emphasize the complementary value and put larger weights on the mid-sized and larger coalition sizes tend to be more replication-robust. To further illustrate this effect, please refer to Appendix A.6 for a discussion on the **binomial semivalues**.

3.4 Application: Machine Learning Data Markets

In this section, we apply our theoretical results to an emerging ML application – data valuation in a multiagent ML data market. Typically, training ML models requires large volumes of high-quality training data. As a consequence, many ML models are trained on standard benchmark problems using carefully collected public datasets such as ImageNet [30] or the UCI Machine Learning Repository [31]. However, when training ML models for real-world applications there is often an under-supply of high-quality public data as these ML applications may require specialized, up-to-date, labeled training data of sufficient volume and fine-grained categories [128]. Consequently, obtaining such training data can be a critical bottleneck in ML [110], which typically relies on data acquisition methods such as searching, manual collection, crowd-sourced labelling, etc. In many cases, multiple potential data providers can be identified, and the data would be acquired separately from each data provider. This motivates us to develop ML data markets that readily connect data collectors (buyers) with data providers (sellers) and accurately valuate data. [2, 63].

A naive implementation of such a market in the form of direct data exchange is likely to fail in practice due to the following reasons: *(i)* Data can be *freely replicated*, and hence can be easily resold by a buyer. *(ii)* Acquiring ownership of a large dataset may exceed the budget of the buyer. Both of these issues can be alleviated by considering data exchange as an integral part of an ML platform. This platform could bring together data from multiple sellers and return an ML model trained specifically for the application of the buyer. The buyer will then pay a fee according to the performance of the model. A key question then arises in this setting: How can the market allocate the payoff among the data sellers?

The most common game-theoretic payoff-allocation method is the Shapley value. However, as we have shown in the previous sections, the Shapley value can be vulnerable to replication manipulations when applied in such an ML market, as data can be easily replicated and the ML model performance typically exhibits a submodular behaviour. Therefore, a malicious seller can exploit properties of the Shapley value to gain higher payoffs by replicating its data and acting under multiple identities. In the following, we model the market as a submodular game and apply our findings from the previous section to study the effects of replication as well as related attack methods from the malicious player.

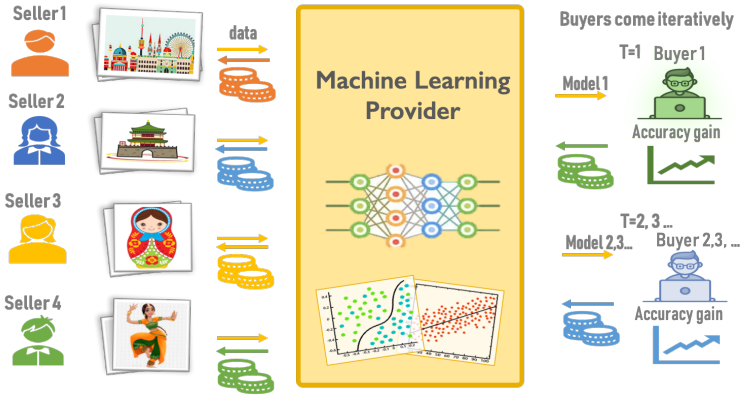


Figure 3.6: Machine Learning Data Market

3.4.1 Data Market as a Submodular Game

Figure 3.6 illustrates a conceptual model of an online data market integrated with an ML platform. In the market, the data sellers upload their data onto the central ML provider platform. At each round of interaction, a buyer will provide an ML classification task, specified by a *validation dataset* D_{val} and meta information. For example, in an image classification task, a trained ML model can predict the class of objects in the input image. The training data from multiple data providers can be pooled for jointly training an ML prediction model $\mathcal{M}(\cup_{i\in N} D_i)$.

We model each round of interaction as a cooperative game $G = (N, v)$, where the players i are the data sellers $N = \{1, \ldots, n\}$, each holding a dataset D_i. A natural characteristic value of a coalition is given by the validation performance $Accuracy(\mathcal{M}, D_{\text{val}})$ achieved by the model $\mathcal{M}$ which is trained on the data held by all players in the coalition:

$$\forall C \subseteq N, v(C) := Accuracy(\mathcal{M}(\cup_{i\in C} D_i), D_{\text{val}})$$

Submodularity is often a good model for approximating properties of this accuracy—the value of additional training datasets typically diminishes with growing data size [66]. Moreover, for many ML models, adding replicated data do not significantly change the model's performance and hence the characteristic function satisfies replication-redundancy (Assumption 3.3.3.1).

3.4.2 Data Replication Attack and Related Attack Models

In the ML data market game, a malicious player may replicate its data and act under multiple false identities. We refer to this manipulation as the data replication attack, where the player aims to

obtain a higher payoff through false name manipulation and replication [102, 2]. More broadly, theoretical aspects of malicious manipulations have been studied in the game theory literature, such as merging/splitting [67, 103], collusion via association/proxy agreements [51], false name manipulations [5]. Moreover, to avoid replica detection, a player may execute more complex manipulations such as adding noise to the replicated data or only replicating subsets of the data. In this section, we will first discuss the naive approach to data replication. Then we will discuss some related attack models. The next definition characterises the naive data replication attack following our replication manipulation defined in Definition 3.3.3.1.

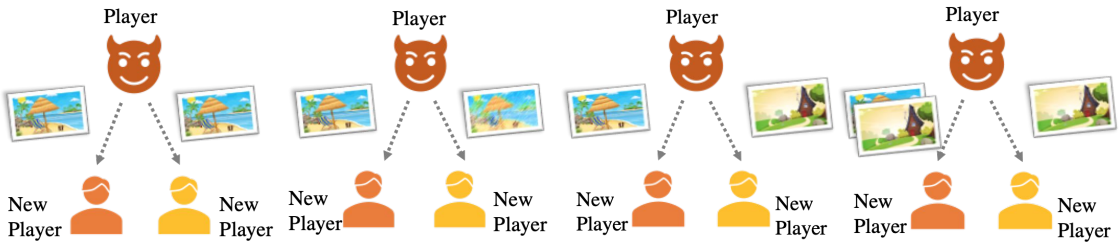
Definition 3.4.2.1 (Data Replication Attack)**.** *In the market game $G = (N, v)$, a malicious player i may execute a replication action k times on its data D_i and act as $k+1$ players $C^R = \{i_0, i_1, \ldots, i_k\}$ each holding one replica of D_i. Denote the induced market game as $G^R = (N^R, v^R)$, where the players $N^R = N \setminus \{i\} \cup C^R$, and the characteristic function v^R satisfies $\forall C \subseteq N \setminus \{i\}\ \forall i_k \in C^R\colon v^R(i_k \cup C) = v(i \cup C)$ and $v^R(C) = v(C)$. By replicating, player i receives a* total payoff *which is the sum of the payoff of itself and all its k replicas, i.e., $\varphi_i^{\text{tot}}(k) = \sum_{\kappa=0}^{k} \varphi_{i_\kappa}(N^R, v^R)$.*

As the market game is an instance of submodular game with replication-redundant characteristic function, we can apply our replication-robustness results in submodular games. For example, from Theorem 3.3.6.2, we can show that in an ML data market game, the Shapley value is vulnerable against the data replication attack, and the malicious player is incentivised to replicate its data and act under multiple identities in order to increase its total payoff. Whereas the Banzhaf value and Leave-one-out are robust against data replication. Similarly, the sufficient and necessary conditions for replication-robustness in Theorem 3.3.6.1 also hold for the data market game.

Despite the fact that replication-redundancy is often a suitable assumption as demonstrated by our empirical results, and can also apply to alternative problem settings (e.g., duplicate features in an ML model input), there may be scenarios where absolute replication-redundancy is not satisfied, for example, data replication which:

1. results in a approximately replication-redundant characteristic function;

2. provides a negative influence on the ML model performance.



(a) Replication **(b)** Perturbation **(c)** Splitting **(d)** Subset Replication

Figure 3.7: Data Replication and Related Attack Models

3. together with data transformations, provide a useful contribution towards the ML model performance, e.g., data augmentation [124].

For the first case (i.e., approximate replication redundancy), we show in the next section that the replication robust solution concepts derived in the previous condition are ϵ-replication robust. For the second case, the total payoff of the malicious player does not exceed the replication-redundant case and hence the replication robust solution concepts under absolute replication-redundancy will still be replication robust. In the third case, our replication-robustness condition can serve to factorise the payoff gain through replication/ transformations into two sources, i.e., 1. *contribution* which comes from the gain in model performance by the replicated data, and 2. *replication*, which only comes from the act of replication. Our replication robustness results studied the second source and thus can be a neutraliser of this effect. Consequently, with the replication robust solution concepts, the replicating player can be rewarded for its useful contribution towards the model performance, rather than the act of replication.

Though the consequences of data replication have been studied in our results for general submodular games, the attack strategy of a malicious player may not be limited to data replication. Potentially more complex attack strategies can be designed as a combination of multiple manipulations. Nevertheless, our framework provides a nice basis for studying these related attack models. In the following sections, we provide some examples which generalise our data replication attack and discuss their relations.

3.4.2.1 Perturbed Replication

After replicating into multiple identities, a malicious player may further perturb each of its replicated datasets to avoid replica detection, as illustrated in Figure 3.7b, such as adding noise

or other data transformation processes such as translation or rotation. We now show that the difference between the payoff of a perturbed replica and a non-perturbed replica is a function of the marginal contributions induced by the perturbation.

Perturbed replications can be formulated as follows: In the market game $G = (N, v)$, a malicious player i replicates its dataset D_i k times and acts as $k+1$ players $C^R = \{i_0, \ldots, i_k\}$ where $D_{i_k} = D_i$. The player can further perturb its replicas as $C^P = \{p_0, \ldots, p_k\}$, where $D_{p_k} = f_k(D_i)$ for some perturbation function f_k, e.g., adding noise. The malicious player receives a total payoff as a sum of all its perturbed replicas, i.e., $\varphi_i^{\text{replicate}} = \sum_{i_k \in C^R} \varphi_{i_k}$ and $\varphi_i^{\text{perturb}} = \sum_{p_k \in C^P} \varphi_{p_k}$. Assume the effect of perturbations are small such that the marginal contribution of the perturbed replicas towards other players' data remain unchanged from the original replica, that is:

$$\forall_{C \subseteq N \setminus \{i\}}, \mathcal{MC}_{p_k}(C) = \mathcal{MC}_{i_k}(C),$$

and the marginal contributions of each perturbed replica towards coalitions containing other perturbed replicas are small, compared to their marginal contributions towards coalitions of other players, i.e, $\exists \epsilon$, s.t. $\forall C \subseteq N \setminus \{i\}, \epsilon \ll \mathcal{MC}_{p_k}(C)$, such that

$$\forall p_k \in C^P, p_j \in C^P \setminus \{p_k\}, \mathcal{MC}_{p_k}(\{p_j\} \cup C) \leq \epsilon$$

Lemma 3.4.2.1. *Compared with replication, the additional gain in total payoff of the malicious player due to the perturbation when replicating k times is given by:*

$$\varphi_i^{\text{perturb}} - \varphi_i^{\text{replicate}} \leq (k+1)\epsilon.$$

Proof. For the detailed derivation please refer to Appendix A.7. □

On the one hand, perturbations which yield negligible marginal values towards other players and the other perturbed replicas (i.e., ϵ is negligible) will yield negligible benefit compared with the non-perturbed replicas. In this case, the replication-robust solution concepts are also $\epsilon-$robust against the perturbed replication attacks. On the other hand, if the perturbation make significant contribution towards the prediction tasks, the perturbed data can be considered as data augmentation, which would be encouraged and naturally reflected in the total payoff.

3.4.2.2 Data Splitting and Replicating Subsets of Data

In addition to perturbation methods such as adding noise, a malicious player may also choose to replicate a subset of its data and act under false identities. We refer to this attack method as *subset replication*, which is illustrated in Figure 3.7d. Alternatively, a player may split its data and acts as multiple identities. We refer to this attack method as *splitting*, which is illustrated in Figure 3.7c. Data splitting also relates to a line of prior work on splitting and merging manipulations, and the discussion of the related work on splitting and merging can be found in the related work section (Section 3.2). In this section, we discuss some relations among replication, splitting and subset replication in terms of the payoff gain for the malicious player. We demonstrate that under payoff allocation using the Shapley value, our considered replication attack (which we refer to as *replication* and illustrated in Figure 3.7a) yields the highest payoff for any $k \geq 1$ (Lemma 3.4.2.2). On the other hand, for the Banzhaf value, the three attacks yield equal payoff when $k = 1$ (Lemma 3.4.2.3).

The related attack models can be formulated as follows: In the market game $G = (N, v)$, the malicious player i performs one of the following three attacks and carries the identities $C^R = \{i_0, \dots i_k\}$: (1) Under replication, each identity holds data $D_{i_k^{\text{rep}}} = D_i$. The total payoff is $\varphi_{\text{tot}}^{\text{rep}} = \sum_{i_k^{\text{rep}} \in C^R} \varphi_{i_k^{\text{rep}}}$; (2) Under splitting, each identity i_k^{split} holds a disjoint subset of the data $D_{i_k^{\text{split}}} \subseteq D_i, \forall j \neq k, D_{i_k^{\text{split}}} \cap D_{i_j^{\text{split}}} = \emptyset$; and the total payoff of the malicious player is $\varphi_{\text{tot}}^{\text{split}} = \sum_{i_k^{\text{split}} \in C^R} \varphi_{i_k^{\text{split}}}$; (3) Under subset replication, each identity i_k^{sub} holds a subset of the data $D_{i_k^{\text{sub}}} \subseteq D_i$ while $D_{i_0^{\text{sub}}} = D_i$, and the total payoff of the malicious player is $\varphi_{\text{tot}}^{\text{sub}} = \sum_{i_k^{\text{sub}} \in C^R} \varphi_{i_k^{\text{sub}}}$.

Lemma 3.4.2.2. *Under payoff allocation using the Shapley value, the total payoff to a malicious player by replication is no less than subset replication and splitting.*

Lemma 3.4.2.3. *Under payoff allocation using the Banzhaf value, executing one of the three aforementioned attack actions and acting under two identities yields equal payoff.*

Proof. The detailed proof for the above lemmas can be found in Appendix A.7 □

In summary, we have provided three example cases of attack models in relation to our replication model. In the future, our framework can be extended to study more related attack models.

3.4.3 Efficient Computation

Due to the nature of semivalues, computing the payoff allocation requires the characteristic values of all possible coalitions, which is exponential on the number of players and often very computationally expensive. For more efficient computation, we may sometimes find closed-form solutions. For example, in Section 3.3.2, we derived closed-form solutions which gave rise to efficient algorithms for computing the Shapley and the Banzhaf value in the facility location game. However, in more real-world applications such as the ML data market, a well structured characteristic function is usually absent and hence no closed-form solutions to these semivalues are available. Moreover, the computation of the semivalues in the such ML applications is very demanding, as it requires the evaluation of $v(\cdot)$ for a large number of coalitions, each involving the training or evaluations of the ML models. As this quickly becomes infeasible, methods for approximating the solution concepts are crucial. In this section, we draw inspirations from prior sampling methods, and introduce a sampling algorithm which can estimate the semivalues.

3.4.3.1 Prior Works

Prior approximation methods [35, 6] mainly focus on the Shapley value and often investigate specific game structures such as weighted voting games. In the following we list a few of these important approximation methods that are relevant to our problem setting.

Random (Permutation) Sampling [18, 2]. This is one of the most well-known methods for approximating the Shapley value. In particular, the method samples T permutations π^t of players, iteratively computes the marginal contribution $\mathcal{MC}_i^t$ of each player i towards the preceding players in π^t, and approximates the Shapley value of each player by averaging over the samples $\hat{\varphi}_i^{\text{shapley}} = \frac{1}{T}\sum_t \mathcal{MC}_i^t$.

Stratified Sampling [86]. This method is a stratified approach that approximates for each player the average marginal contributions $z_i(c)$ over size-c coalitions, and then computes the Shapley value by averaging over $z_i(c)$'s. Though the original paper [86] focused on the Shapley value, we observe that this method can also be extended to be used for general semivalues.

Group Testing [63]. This is a more recent method developed for approximating the Shapley value. In particular, this method effectively shares sampled coalitions among the players. In each turn t, it draws a coalition C_t of a sampled size k_t. After drawing the coalitions, the method

Algorithm 5 Sampling Algorithm for Semivalues

Input: T: number of samples, N: the set of players, $q(c)$: a sampling distribution over coalition sizes c, α_c, importance weights of the semivalue φ.
Output: estimated payoff value $\hat{\varphi}'_i$ of each player

Step 1: Sampling coalitions and computing utilities
for $t = 1, 2, \ldots T$ **do**
    Draw coalition size $c_t \sim q(c)$
    Uniformly sample coalition $C_t \subseteq N$ of size c_t
    Compute $v(C_t)$
end for
$U_c \leftarrow \frac{1}{||C_t|=c|} \sum_{|C_t|=c} v(C_t)$ for all c
$\bar{U}_i(c) \leftarrow \frac{1}{||C_t|=c, i\in C_t|} \sum_{|C_t|=c, i\in C_t} v(C_t)$ for all c and i
Step 2: Approximating players' values
$\hat{\varphi}_i \leftarrow \sum_{c=0}^{N-1} \alpha_c \frac{N}{N-c} (\frac{N-c}{N} \bar{U}_i(c+1) + \frac{c}{N} \bar{U}_i(c) - U_c)$
Compute players' pairwise differences $\Delta\hat{\varphi}_{ij} \leftarrow \hat{\varphi}_i - \hat{\varphi}_j$,
Compute approximated total payoff $\hat{\varphi}_{\text{all}} \leftarrow N \sum_{c=0}^{N-1} \alpha_c (U_{c+1} - U_c)$
Find $\hat{\varphi}'_i$ by solving a feasibility program with the following constraints:
$\sum_i \hat{\varphi}'_i = \hat{\varphi}_{\text{all}}$, $|(\hat{\varphi}'_i - \hat{\varphi}'_j) - \Delta\hat{\varphi}_{ij}| \leq \epsilon$ for $\forall i, j \in N$

proceeds by estimating players' pairwise differences $\Delta\hat{\varphi}_{ij} \propto \sum_t (\mathbb{1}_{i\in C_t} - \mathbb{1}_{j\in C_t}) v(C_t)$. Then, by using the sum of their payoffs, which is the value of the grand coalition for the Shapley value $v(N)$, the approximate Shapley value can be obtained for each player by solving a feasibility program.

Among the above methods, random permutation sampling and group testing cannot be used to approximate solution concepts beyond the Shapley value. In particular, the group testing approach requires knowledge of the total allocated payoff. This can only be efficiently obtained for the Shapley value as the value of the grand coalition $v(N)$ due to the Efficiency Axiom **(A2)**. Stratified sampling can be extended beyond the Shapley value through Equation 3.3, however, with a growing number of players, separate assignment of samples to players can be suboptimal.

3.4.3.2 A Sampling Algorithm for Semivalues

Motivated by the above considerations, we propose Algorithm 5, a sampling algorithm which can provide estimations for general semivalues. By rewriting the players' average marginal contributions, our algorithm can improve the sample efficiency of stratified sampling by enabling sample sharing among the players. Moreover, compared with group testing, our algorithm approximates the total allocated payoff, hence can extend beyond the Shapley value to general semivalues.

The algorithm is based on two unbiased estimators for the payoff of each player and the total payoff for all players, which are presented in the next theorem.

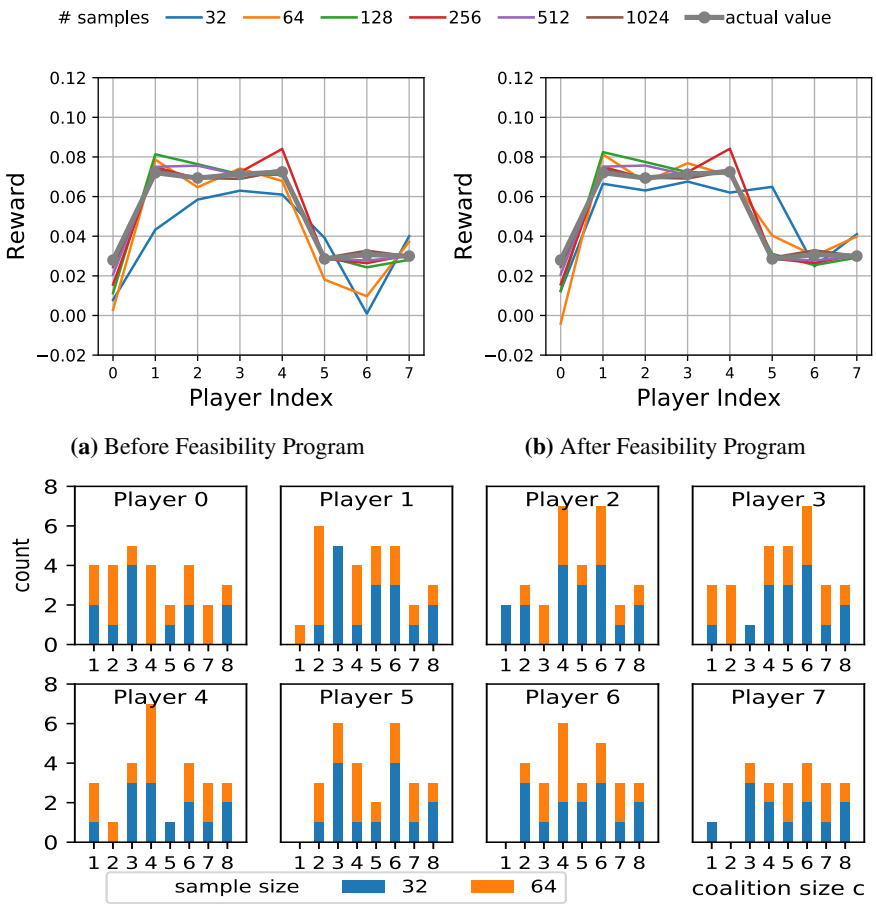
Theorem 3.4.3.1. *Let* $\varphi_i = \sum_{c=0}^{|N|-1} \alpha_c z_i(c)$ *denote a semivalue. Algorithm 5 uses unbiased estimators* $\hat{\varphi}_i = \sum_{c=0}^{|N|-1} \alpha_c \frac{|N|}{|N|-c}[\frac{|N|-c}{|N|}\bar{U}_i(c+1) + \frac{c}{|N|}\bar{U}_i(c) - U_c]$ *for the payoff of each player i, and* $\hat{\varphi}_{\text{all}} = N\sum_{c=0}^{|N|-1} \alpha_c(U_{c+1} - U_c)$ *for the total payoff allocated to all players, where* U_c *is the mean of size c coalitions, and* $\bar{U}_i(c)$ *is the mean of size c coalitions which contain player i.*

Proof. For the proof of the unbiasedness of the estimators please refer to Section A.8 □

Algorithm 5 iterates for T steps, and for each step, it samples a coalition of coalition size c according to an input distribution $q(c)$, and evaluate the coalition value through ML model training. Then the algorithm approximates the payoff of each player according to the estimator from Theorem 3.4.3.1, that is, $\hat{\varphi}_i = \sum_{c=0}^{|N|-1} \alpha_c \frac{|N|}{|N|-c}(\frac{|N|-c}{|N|}\bar{U}_i(c+1) + \frac{c}{|N|}\bar{U}_i(c) - U_c)$, where $\bar{U}_i(c)$ is the sample mean of sampled size-c coalitions' values which contain player i, and U_c is the sample mean of all sampled coalitions of size c. Next, we compute the players' pairwise differences $\Delta\hat{\varphi}_{ij}$, and the total allocated payoff according to the estimator from Theorem 3.4.3.1, that is, $\hat{\varphi}_{\text{all}} \leftarrow N\sum_{c=0}^{N-1} \alpha_c(U_{c+1} - U_c)$. Finally, we compute the estimated payoff of each player $\hat{\varphi}'_i$ through a feasibility program which makes use of the players' pairwise differences and the approximated total allocated payoff. The feasibility program is formulated as follows, where ϵ is a small tolerance value. The program can be solved using software such as the Python PuLP package [89].

$$
\begin{aligned}
\underset{\hat{\varphi}'_1,\ldots,\hat{\varphi}'_{|N|}}{\text{minimize}} \quad & 0 \\
\text{subject to} \quad & \sum_{i=1}^{N} \hat{\varphi}'_i = \hat{\varphi}_{all}, \\
& \hat{\varphi}'_i - \hat{\varphi}'_j - \Delta\hat{\varphi}_{ij} \leq \epsilon, \qquad i,j = 1,\ldots,|N| \\
& \hat{\varphi}'_i - \hat{\varphi}'_j - \Delta\hat{\varphi}_{ij} \geq -\epsilon, \qquad i,j = 1,\ldots,|N|
\end{aligned}
$$

The feasibility program is inspired by the group testing approach. In the small sample regime, that is when the samples provide insufficient coverage of the different coalition sizes, the feasibility program can help adjust the estimates by adding a sum constraint over the payoffs to the players and thereby adjusting the values among the players.



(a) Before Feasibility Program

(b) After Feasibility Program

(c) Count of each player in the sampled coalitions

Figure 3.8: Example effect of applying the feasibility program.

Figure 3.8 illustrates the effect of the feasibility program for approximating the Shapley value of an 8 player ML market game. More details about the market games are included in the experiments section (Section 3.5). Specifically, Figures 3.8a and 3.8b compare the Shapley values of each player which are estimated using Algorithm 5, before and after applying the feasibility program. In Figure 3.8a, the estimated Shapley values of the players with 32 samples (blue line) largely deviates from their true Shapley values (grey line). After applying the feasibility program, the players' estimated values are improved due to adjustment among each other's values, as shown in Figure 3.8b (blue line). Moreover, Figure 3.8c shows the coverage of the sampled coalition sizes for each player for sample size 32 and 64. In particular, for sample size of 32, some coalition sizes are not covered, and adjustments using the feasibility program helps reduce the negative effect.

The sampling algorithm significantly reduces the number of coalition value evaluations from $2^{|N|}$ to the number of samples, and all other calculations incur negligible time. Orthogonal to

a sampling based approach, approximations to the characteristic values $v(C)$ may be applied to further improve efficiency, and would be an interesting direction for future work.

3.5 Experiments

In this section, we provide empirical validations to our theoretical results. Specifically, we will first discuss the replication-robustness of the Shapley value and the Banzhaf value in the facility location game computed using Algorithm 4. Then we present results on the ML data market: we will present experiments that demonstrate the submodularity and replication-redundancy of the characteristic function. We then compare the replication-robustness properties of the discussed solution concepts in the ML data market game. Finally, we demonstrate that our sampling algorithm shows a significant increase in sample efficiency over the baseline methods.

3.5.1 Experimental Setup

Facility Location Game. To generate the facility location games, we sample the utility matrix (c.f., Definition 3.3.2.2) U_{ij} for $|D| = 10$ (and $|D| = 20$) customers and a varying number of players (which are the facility locations), where each utility value u_{ij} is an integer that is uniformly sampled within the range of $[0, 20]$. To compare the replication-robustness of the Shapley value and the Banzhaf value, we compute the total payoff of the malicious player with respect to increasing number of replications. Specifically, we start with a facility location game of $|\mathcal{L}| = 10$ players including one malicious player i, where the game is generated following the above procedures. Then we gradually increase the number of replications k from 0 to 50, and plot the player i's total payoff in each of the induced games.

The ML Data Market Game. For the data market game, we adopt three standard ML datasets of varied input sizes and assign to each player a subset of the data. One of the players is malicious which replicates its data and gradually increases the number of replications k. We next introduce the training datasets, followed by the ML models and training procedures.

(a) *Covertype* [31]: Each input consists of 10 continuous features, and the output is a prediction of the forest cover type out of 7 classes. We use the dataset provided by Kaggle which consists of ~15000 training datapoints uniformly distributed in the 7 output classes. In our experiments, 5 honest players each holds 1000 datapoints, 5 replica players share 1000

Table 3.1: Experimental settings for CIFAR-100: training and validation data assignments. The *Uniform, Disjoint, Mixed* experiments use all 20 superclasses and their 100 subclasses. Players 1-4 are honest players while 5-7 are replicas that hold the same data as malicious player 0.

	N	Subclass C_{sub}	Training Data Assignment	Validation Datasets
Uniform	8	100	Players $0-4$ assigned data from 100 C_{sub} uniformly Players (replicas) $5-7$ assigned the same data as Player 0	All C_{sub}s' validation data
Disjoint	8	100	Players $0-4$ each assigned 20 C_{sub} Players (replicas) $5-7$ assigned the same data as Player 0	All C_{sub}'s validation data
Mixed	8	100	Players $0-4$ assigned varied portions of each C_{sub} Players (replicas) $5-7$ assigned the same data as Player 0	All C_{sub}s' validation data

datapoints. The 10 continuous features are (elevation, aspect, slope, horizontal distance to hydrology, vertical distance to hydrology, horizontal distance to roadways, hillshade 9am, hillshade noon, hillshade 3pm, horizontal distance to fire points). The preprocessing steps include normalization of each field to $[0, 1]$.

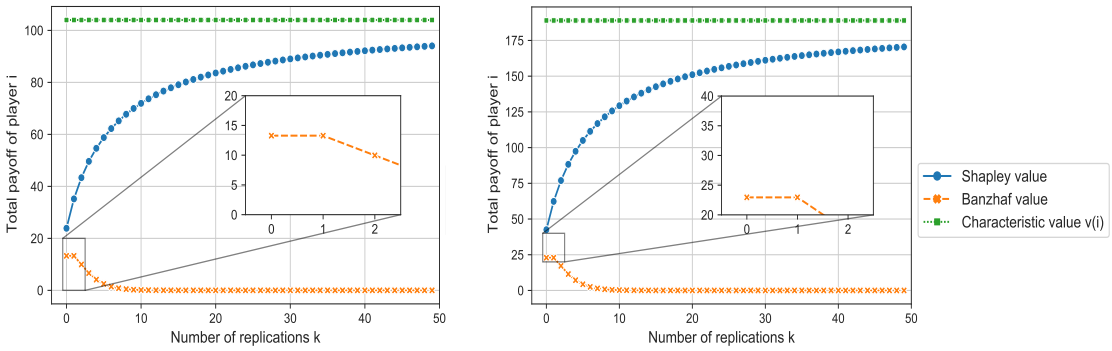
(b) *CIFAR-100* [70]: The input features are 32x32x3 RGB images of 20 superclasses (e.g., trees, vehicles) and 100 subclasses (e.g., maple, oak, bicycle, bus). The output is a prediction of the subclass out of the 100 possible subclasses. Preprocessing steps include normalizing the image matrices to [0,1]. We carried out experiments with varied data assignments (i.e., *Uniform, Disjoint, Mixed*), and the detailed data assignments are provided in Table 3.1.

(c) *Tiny ImageNet* [74]: The input features are 64x64x3 RGB images which belong to 20 random classes, and the output is the class. In our experiments, 3 honest players each holds 2000 different datapoints and 3 replicas share the same 2000 datapoints.

Models and Training. For the Covertype classification task, we use a 4-layer fully-connected neural network with 512 units per layer. The model is trained for 20000 steps using the Adam optimiser [64], learning rate of 0.0001 and minibatch size of 128. For the CIFAR-100 experiments, we used the VGG-16 architecture [126] with 10 convolutional layers of kernel size 3 and max-pooling, followed by 2 fully-connected layers with 1024 units per layer. The model is trained for 15000 steps, using the learning rate of 0.001 and minibatch size of 64. For Tiny ImageNet, we again use the VGG-16 network with the same configurations for the convolutional layers, and 2 fully-connected layers each of 4096 units. The model is trained for 10000 steps using the learning rate of 0.001 and minibatch size of 64.

Table 3.2: Efficient Computation of the Shapley value and the Banzhaf value in the facility location game. The table compares the time (in seconds) used for computing the exact Shapley and Banzhaf value for all players using the standard and fast algorithms. *n* refers to the varying number of players. *standard* refers to an algorithm that enumerates all possible coalitions and computes the marginal contributions, while *fast* refers to Algorithm 4. The entries in bold refer to the best for the semivalue for the number of players.

		n=10	**n=15**	**n=20**	**n=50**	**n=100**
Shapley value	*standard*	0.283	14.182	558.525	-	-
	fast	**0.004**	**0.007**	**0.011**	**0.099**	**0.225**
Banzhaf value	*standard*	0.245	12.623	482.803	-	-
	fast	**0.002**	**0.003**	**0.006**	**0.056**	**0.170**



(a) $|\mathcal{L}| = 10, |D| = 10$ **(b)** $|\mathcal{L}| = 10, |D| = 20$

Figure 3.9: Replication-robustness of the Shapley and Banzhaf value for the Facility Location Game. The characteristic value of $v(i)$ is a single value, which is invariant across k and is shown to visualise the convergence behaviour of the Shapley value.

3.5.2 Results

We first compare the replication-robustness of the Shapley value and the Banzhaf value on the facility location game. Then we present the results on the ML data market experiments for the submodularity assumption, replication-robustness, and efficient sampling.

3.5.2.1 Facility Location Game

Table 3.2 demonstrates our Algorithm 4 (fast) for computing the Shapley value and the Banzhaf value in the facility location game using our closed-form solutions, compared with the standard algorithm (standard) by enumerating all possible coalitions and computing the marginal contributions. The results computed by the fast algorithm are verified to be equal to the standard algorithm, and the table compares their computation time run on a Macbook laptop computer. From the table we can observe that the standard algorithm struggles in games of a large number of players (e.g., when $n = 50$) while Algorithm 4 can scale up easily.

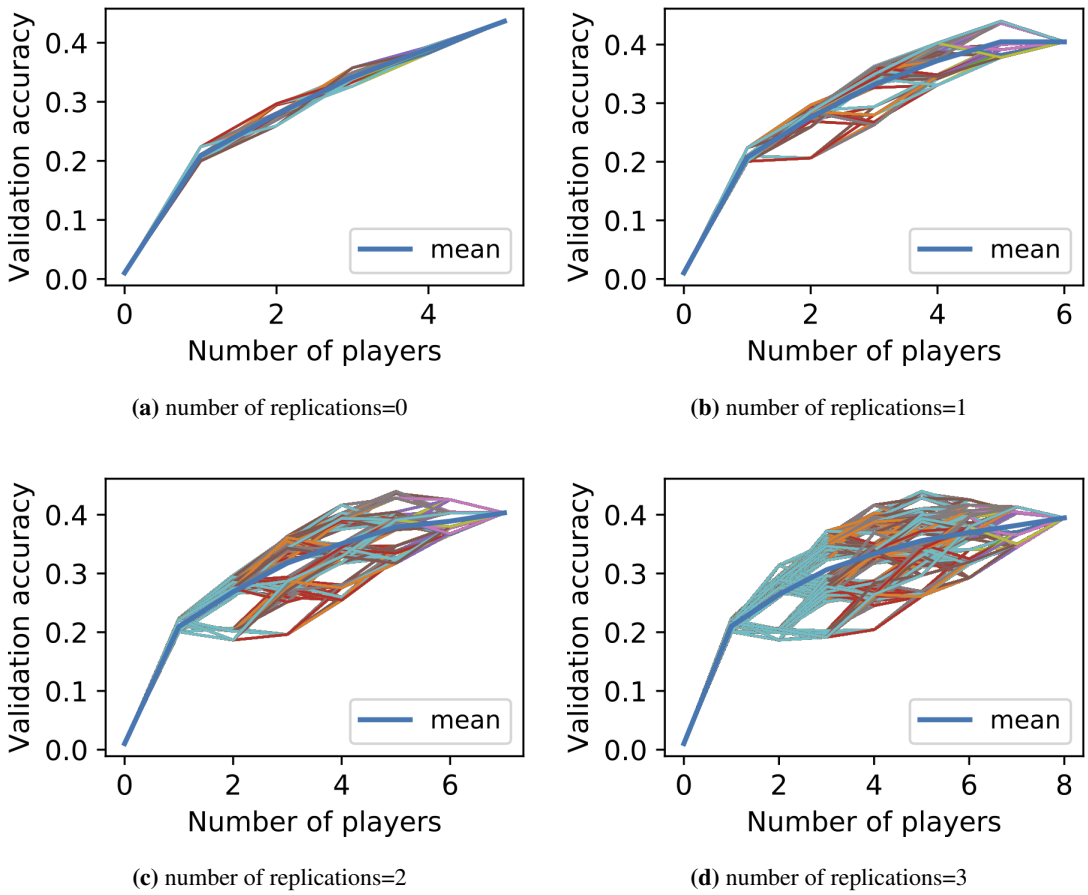
With the help of Algorithm 4, we are able to efficiently compute the payoff changes of a replicating player over a large number of replications. This allows us to visualise the convergence properties of the Shapley value and the Banzhaf value under replication, and compare their replication-robustness properties, as shown in Figure 3.9. Specifically, the original game includes (Figure 3.9a) $|\mathcal{L}| = 10$ players, $|D| = 10$ customers ; (Figure 3.9b) $|\mathcal{L}| = 10$ players, $|D| = 20$ customers. In both cases, among the players, player $i = 0$ replicates itself k times and act under $k + 1$ identities, and receives the total payoff of all the replicas. The curves show the total payoff of the replicating player with respect to the growing number of replications k. The graph validates some of our theoretical results: (1) The Shapley value is not replication-robust, and the total payoff of the player monotonically increases and converges to its characteristic value (green line). (2) Moreover, the unit gain of the player for adding more replicas monotonically decreases. This can be seen from the decreasing height differences between the neighbouring points. (3) By contrast, the Banzhaf value is replication-robust. The total payoff of the replicating player monotonically decreases and converges to 0. (4) Moreover, by comparing $k = 0$ and $k = 1$ (c.f. the zoomed region), we can see that the Banzhaf value is neutral to the first replication where the total payoff of the replicating player is unchanged.

Next, we present the replication-robustness results for the ML data market game. Before that, we will first empirically demonstrate our assumptions, i.e., Assumption 3.3.1.1 on *submodularity* and Assumption 3.3.3.1 on *replication redundancy*.

3.5.2.2 Properties of the Data Market Characteristic Function

In Figure 3.10, we show the accuracy curves of all possible permutations of the players on the CIFAR-100 Uniform experiment. The detailed descriptions of the plots can be found in the caption. The subplots show the varying number of replicas. By comparing Figures 3.10a - Figures3.10d, the property of replication-redundancy can be observed: as more replicas are added, there is a growing number of approximately horizontal lines in the plots. The mean curves (thick blue line) shows the mean of all curves in each plot which display a submodular behaviour.

Moreover, the submodularity and replication-redundancy properties can be observed from the marginal contributions. Figure 3.11 shows the average marginal contributions $z_i(c)$ of each player towards coalitions of sizes c across all three ML datasets. We can observe that in all subplots



(a) number of replications=0

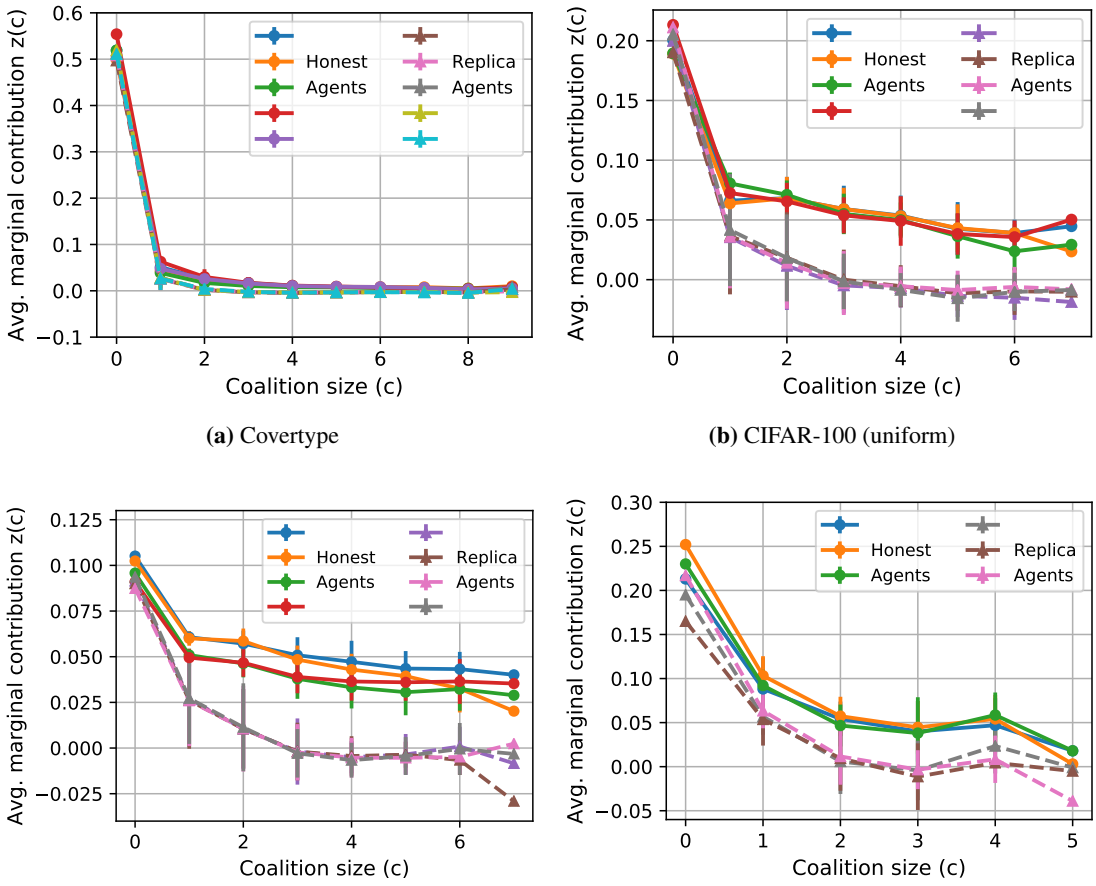
(b) number of replications=1

(c) number of replications=2

(d) number of replications=3

Figure 3.10: Validation accuracy vs. the number of players across a varying number of replications on CIFAR-100. (a) is the original game with no replicas, and (b)-(d) adds 1-3 replicas, respectively. Each graph plots all permutations of the players, where each curve represents a specific permutation. Along the x-axis, we start from the empty coalition, and the players join in the order according to the permutation. Each point in the curve shows the accuracy of the ML model trained over data pooled from the players that have joined.

which represent different ML datasets, $z_i(c)$ is monotonically decreasing. This is a result of the submodularity of the ML model accuracy function (c.f. Lemma 3.3.5.1). The curves further demonstrate replication-redundancy, where $z_i(c) \approx 0$ for the replica players when c exceeds the total number of honest players. Moreover, the standard deviations (c.f. the error bars) of $z_i(c)$ are high for the replica players. This is due to the large differences in the replica players' marginal contributions when joining coalitions with/without another replica. Occasionally the marginal contribution of a replica towards coalitions containing other replicas becomes negative, which is a result of the randomness that occurs during training.



(a) Covertype

(b) CIFAR-100 (uniform)

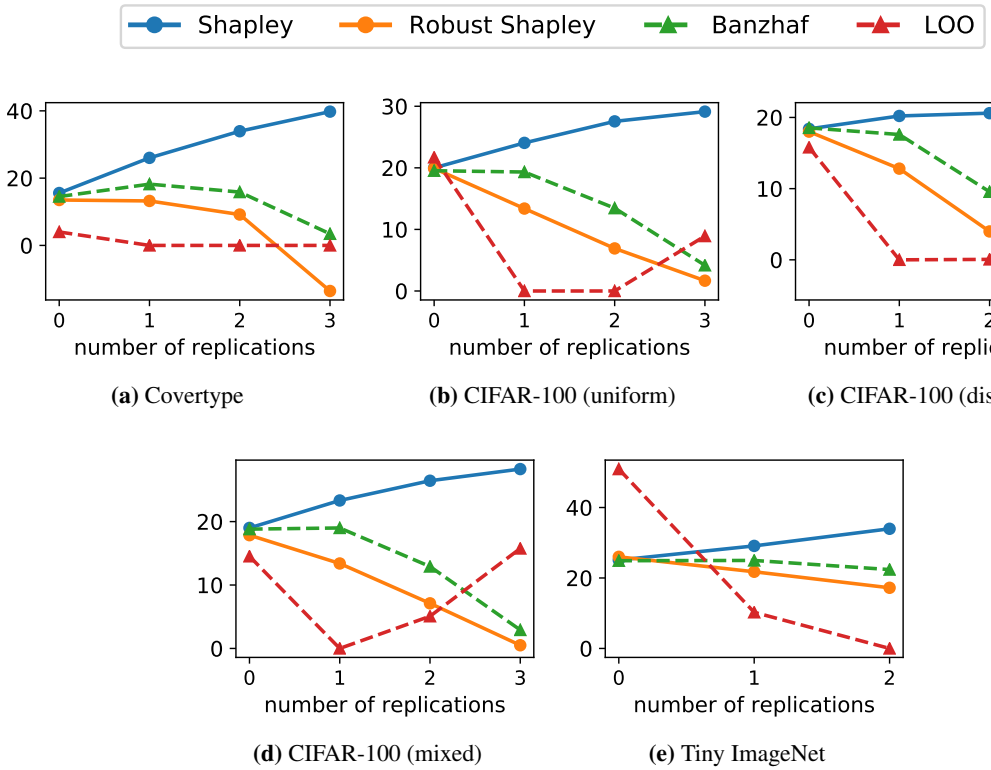
(c) CIFAR-100 (disjoint)

(d) Tiny ImageNet

Figure 3.11: Average marginal contributions $z_i(c)$ of all players across various datasets. Solid lines are the honest players while dashed lines are replicated identities which belong to the malicious player. Error bars show standard deviations of the marginal contributions of each coalition size. We can observe that $z_i(c)$ decrease monotonically with coalition size, which is a result of submodularity. Moreover, $z_i(c) \approx 0$ for the replica players when c exceeds the total number of honest players due to replication-redundancy.

3.5.2.3 Replication-Robustness

Figure 3.12 compares the replication-robustness properties of various important solution concepts across the different ML datasets. The curves show the change in total payoffs of the replicating player as a percentage of the total allocated payoffs, over the growing number of replicas. Along the x-axes, the Covertype, CIFAR-100, and Tiny ImageNet tasks start with 5,4,3 honest players respectively and another one malicious player, then we gradually increase the number of replicas. In all settings, we can observe that the Shapley value is vulnerable to replication, and the total share of value gained by the malicious player increases. Both the Banzhaf value and the Robust Shapley value are replication-robust. The Leave-one-out value only includes a player's marginal



(a) Covertype **(b)** CIFAR-100 (uniform) **(c)** CIFAR-100 (disjoint)

(d) CIFAR-100 (mixed) **(e)** Tiny ImageNet

Figure 3.12: Percentage of total replica values in the total allocated payoffs w.r.t. number of replications. Along the x-axis, we increase the number of replications by the malicious player, e.g. x=3 refers to an induced game where the malicious player holds 4 replicas.

contribution towards all other players, hence can be sensitive to the randomness which occurs during training. The percentage is plotted for easy comparison, which also preserves the trend of the actual values. The curve for the Leave-one-out (LOO) value exhibit random behaviours because the LOO value is the marginal contribution of a player towards all other players. Therefore it exhibits a near-zero absolute value and hence is very susceptible to randomness during training.

3.5.2.4 Sampling Algorithm

We compare the performances of our sampling algorithm against baseline approaches, namely, *random sampling, stratified sampling, and group testing*. We run the algorithms to approximate the Shapley value, Banzhaf value and Robust Shapley value across all three ML datasets and an additional simulated monotonic random set function that contains 10 players of varied importance. In addition, we test the algorithms on two facility location games, where the actual values of the Shapley value (128 players) and the Banzhaf value (64 players) can be efficiently computed using Algorithm 4. The entries to the utility matrix are real values that are sampled uniformly from

Algorithm 6 Random Set Function

Input: N: the set of all players
Output: $v : 2^N \to \mathbb{R}$: characteristic function of all possible coalitions of the players

for coalition $C \subseteq N$ **do**
    $\mu \leftarrow 0.01 \sum_{i \in C} i$
    $\sigma \leftarrow 0.05$
    Assign utility to the coalition
    $v(C) \leftarrow (1 - e^{-|C|}) + \mathcal{N}(\mu, \sigma^2)$
end for

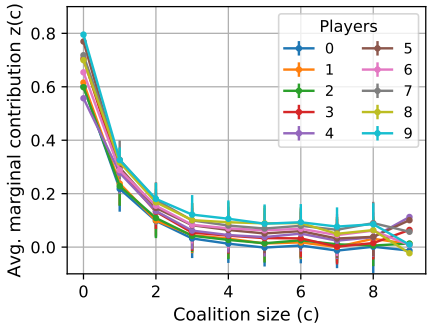


Figure 3.13: Average marginal contributions $z_i(c)$ of player i for the random set function

the range of $[0, 1]$. The simulated random set function is defined as follows:

$$v(C) \coloneqq (1 - e^{-|C|}) + \mathcal{N}(\mu, \sigma^2), \text{ where}$$

$$\mu = 0.01 \sum_{i \in C} u_i, \quad u_i = i, \quad \sigma = 0.05$$

We assign the utility values to all possible coalitions iteratively, according to Algorithm 6. Each player is given a weight u_i proportional to its index, and we add noise to coalition utility $1 - e^{-|C|}$ according to the sum of weights of its member players. The average marginal contributions of each player across coalition sizes $z_i(c)$ is shown in Figure 3.13 which display monotonically decreasing.

Table 3.3 shows results for the sampling algorithms. We present the mean and standard deviation of the *relative error* $\eta_{error} = |1 - \frac{v_{\text{approx.}}}{v_{\text{actual}}}|$ of each algorithm across 10 random seeds. Empty entries (-) indicate that the algorithm cannot be applied to approximate the solution concept, and φ_{all} are relative errors on the estimated total allocated payoff. Compared with all baselines, our algorithm significantly reduces the relative errors and standard deviations on all tasks and datasets.

3.6 Conclusions and Future Work

In this chapter, we explored the intersection between submodular functions, cooperative games and multiagent ML systems, and we looked at two specific submodular games, that is, the facility location game and the ML data market game. For the motivating example of the facility location game, we found closed-form solutions for the Shapley and Banzhaf value which allow for efficient computation (Algorithm 4). We also observed a distinction between the Shapley and Banzhaf values for evaluating the importance of facility locations. With this observation, we looked into general submodular games and showed that the average marginal contributions of players

Table 3.3: Relative error of the sampling algorithms. Bold font indicates best result for each dataset and number of samples.

		Random Sampling		Stratified Sampling		Group Testing		Ours	
	# samples	***64***	***128***	***64***	***128***	***64***	***128***	***64***	***128***
Covertype	*Shapley*	0.93 ± 0.16	0.63 ± 0.11	1.30 ± 0.86	0.89 ± 0.32	2.98 ± 0.26	2.04 ± 0.46	**0.06±0.02**	**0.04±0.01**
	Banzhaf	-	-	0.66 ± 0.23	0.53 ± 0.10	-	-	**0.31±0.12**	**0.25±0.11**
	Robust Shapley	-	-	0.41 ± 0.20	0.30 ± 0.12	-	-	**0.24±0.07**	**0.23±0.08**
CIFAR-100	*Shapley*	0.23 ± 0.12	0.16 ± 0.06	0.30 ± 0.14	0.22 ± 0.11	1.08 ± 0.44	0.67 ± 0.28	**0.13± 0.05**	**0.08± 0.03**
	Banzhaf	-	-	0.31 ± 0.24	0.27 ± 0.25	-	-	**0.20± 0.10**	**0.10± 0.04**
	Robust Shapley	-	-	0.26 ± 0.15	0.16 ± 0.08	-	-	**0.19± 0.09**	**0.11± 0.04**
Tiny ImageNet	*Shapley*	0.38 ± 0.14	0.24 ± 0.12	0.44 ± 0.19	0.24 ± 0.15	1.34 ± 0.39	0.91 ± 0.29	**0.12±0.06**	**0.09±0.03**
	Banzhaf	-	-	0.63 ± 0.39	0.53 ± 0.22	-	-	**0.42±0.12**	**0.26±0.09**
	Robust Shapley	-	-	0.88 ± 0.46	0.67 ± 0.35	-	-	**0.60±0.16**	**0.38±0.16**
Random Set Function	*Shapley*	0.46 ± 0.11	0.36 ± 0.07	0.40 ± 0.21	0.21 ± 0.15	2.00 ± 0.35	1.40 ± 0.33	**0.12±0.02**	**0.08±0.02**
	Banzhaf	-	-	1.06 ± 0.45	0.69 ± 0.20	-	-	**0.51±0.50**	**0.23±0.05**
	Robust Shapley	-	-	1.09 ± 0.38	0.83 ± 0.16	-	-	**0.50±0.20**	**0.28±0.09**
	# samples	***256***		***256***		***256***		***256***	
Facility Location	*Shapley* ($N = 128$)	1.57 ± 0.04		0.95 ± 0.06		57.3 ± 6.61		**0.08± 0.002**	
	Banzhaf ($N = 64$)	-		1.59 ± 0.99 (φ_{all} 14.99 ± 0.99)		-		**0.72± 0.10** (φ_{all} **0.40± 0.02**)	

monotonically decrease as a result of submodularity. We then investigated the effect of redundancy and the replication manipulation on the semivalues in submodular games, and showed that the total payoff to the replicating player according to the Shapley value monotonically increases with respect to a growing number of replications, whereas the total payoff to the replicating player according to the Banzhaf value monotonically decreases and is neutral to the first replication. To generalise these results, we found a necessary and sufficient condition that characterises the replication-robustness of semivalues in general for an arbitrary number of replications.

Following our theoretical study, we applied the results to an emerging application of ML data market, and extended our results from data replication to related manipulations such as perturbation. Moreover, we developed a sampling algorithm for the efficient estimation of the semivalues.

The considered data market scenario is a basic model inspired by related literature [2, 102]. It helps us to evaluate the game theoretic payoff allocations, in particular, the robustness (e.g., against replication) and fairness (e.g., symmetry among players). The market scenario can be further improved and extended to more realistic scenarios. For example, some potential future work include: 1. study data markets where the data buyer refuses to upload its validation data due to privacy, 2. the buyer chooses to manipulate its validation dataset in order to reduce the price of the model, 3. multiple data sellers collude to improve their collective payoff, 4. extending the payoff-allocation methods to large-scale and real-world ML data markets, 5. extending the payoff-allocation methods to alternative multiagent ML systems such as ML feature importance evaluations.

4

Semivalues for Credit Assignment in Robotic Control

Contents

4.1 Introduction

So far we have applied semivalues to evaluate players' contributions in multiagent ML systems. This valuation can be a useful tool for analyses and interpretations inside ML systems such as payoff allocation and importance interpretation towards data and features. This type of analysis is usually performed *post-hoc* and does not directly take part in the machine learning process to improve the learning outcome. Nevertheless, the valuation contains rich information which lends itself to natural learning signals for guiding the learning process and model updates. In this chapter, we study an emerging application of multiagent reinforcement learning in robotics, where we will discuss the opportunities and challenges of using the semivalues as learning signals during the reinforcement learning process.

Reinforcement learning (RL) has recently shown many remarkable successes, e.g., in playing Atari and Go at a superhuman level [90, 125]. Recently, there have been wide research interest and significant advances in robotic learning using RL techniques [73, 38]. Unlike classical RL tasks, which have discrete action spaces and underlying state spaces (e.g. Atari and Go), problems in robotics often have high-dimensional continuous states and actions and are often limited by real-world sample budgets [68]. To this end, prior methods in robotic learning have developed RL algorithms capable of performing continuous control [48, 77, 118, 50], and sample-efficient learning methods, such as model-based RL [37, 61], hindsight experience replay [3], and self imitation learning [100].

Despite the success demonstrated by RL methods in continuous control, when applying RL algorithms to robotic applications, it is essential not to overlook real-life physical limits such as sensing noise and communication delays [68]. Specifically, prior RL approaches typically model a robot as a single, fully-centralised agent which observes the full global state consisting of observations from each body/joint and learns an optimal policy that outputs a combined action for all controllers. Though a centralized controller can compensate for the state of the whole system, when deployed in complex, unseen real-life settings, communications of the local observations between body components can be delayed due to physical limitations. Moreover, robots with a centralised controller do not scale up to a large number of controllers and can easily become incapacitated if any sub-component is compromised. In such scenarios, a decentralised control system is desirable, where the robot controllers learn and execute their own policies.

Fortunately, many real-life robots consist of multiple connected links which can be modelled as a multiagent system [148]. To enable coordination among agents with local observations and decentralised controllers, a standard approach is to perform centralized optimisation during training and decentralised operation during execution (CTDE) [42, 83], i.e., the system can benefit from the state of the whole system during training, while acting in a decentralised manner (please refer to Section 2.2.4 for more background on CTDE). Using this framework, multiagent systems with RL agents have demonstrated coordination skills in complex coordination tasks such as Starcraft [114, 108, 150].

Inspired by the above framework, we apply multiagent RL to robotic learning which exhibits high-dimensional, continuous state and action spaces. Specifically, we extend Proximal Policy Optimisation (PPO) [118]—an algorithm widely used in both discrete and continuous control tasks—to a fully cooperative multiagent framework with CTDE (more background on PPO are included in Section 2.2.3). However, as the robot interacts with the environment as a whole, *only a global reward signal for the whole system is available while a reward function which signifies the contribution of each individual agent is absent.*

To address this problem and evaluate the contribution of each agent, we model the multiagent interaction at each timestep as a cooperative game and apply the semivalues for credit assignments to each agent using its marginal contributions towards all possible coalitions of other agents. Thereafter, the credit assigned towards each agent can be used as its reward signal and applied to guide updates on the agent model.

The game-theoretic framework allows us to effectively evaluate the agent-specific reward signal. However, a new challenge arises when computing the semivalues during the multiagent learning process as opposed to performing post-hoc analyses: To compute the credit assignments, *how can we obtain the value of different coalitions of agents, when only the value of the grand coalition is available (i.e., the global reward of the whole system)?* On the one hand, the credit assignment needs to be performed at each step of training. Typically model training can take over millions of steps for RL applications. Consequently, it is impossible to re-run the simulation for each of the 2^n coalitions at each training step. Alternatively, if we simply rely on the generalisability of the models (such as a neural network) to infer the value of the unseen coalitions from the observed grand coalition, we would suffer from highly inaccurate results.

This stems from a common problem in the multiagent robotics control domain, and is caused by the high dimensional continuous state and action spaces.

To address this challenge, we propose a multiagent model-based learning framework for accurately estimating the values of coalitions of agents. Model-based learning has been an important method in RL and especially robotic learning. Commonly model-based RL allows the agent to make use of a model of the environment (e.g. predicting the transition dynamics or rewards), to generate simulated experiences, which augments the experiences collected from interactions with the real environment. In this chapter, we integrate model-based learning into our multiagent system and use the environment model to simulate values of the unseen coalitions. Moreover, the model can be learned from scratch and in conjunction with the agent policies. We demonstrate that this framework greatly improves credit assignment in continuous control tasks and significantly boosts the rewards and sample efficiency.

Finally, we empirically evaluate our proposed methods on MuJoCo robotic continuous control tasks. We demonstrate that our model-based game-theoretic credit assignment leads to significant improvements in rewards and sample efficiency compared to using only the central reward function and the model-free counterparts of the credit assignment methods. In this chapter, we focus on how to enable the application of game-theoretic payoff allocation methods to the multiagent RL system through model-based coalition value estimations. An interesting next step would be to apply our theoretical insights from the previous chapter to submodular multiagent robotic tasks such as formation control.

4.2 Related Work

Multiagent RL. The simplest form of multiagent RL is Independent Q-learning (IQL) [138], where a group of agents each learns on their own and treats other agents as part of the environment. For IQL, the agents frequently face the challenge that the environment appears non-stationary, as other agents are simultaneously learning and updating their policies. To address this issue, the centralised training, decentralised execution framework was first proposed separately by [42] and [83]. This training paradigm allows the agents to access the global information during training and has been a standard approach to recent multiagent RL algorithms since then. [42] proposed an actor-critic algorithm that addresses multiagent credit assignment explicitly using

the counterfactual value of each agent and is applied to domains with discrete action space such as Starcraft. [83] proposed a multiagent policy gradient algorithm where each agent receives a separate reward and uses their critic. This method can apply to competitive settings but does not address the multiagent credit assignment problem. Another line of work using centralised training, decentralised execution, and Q-learning are the implicit credit assignment methods such as [108, 109, 142]. More recently, [150] showed the effectiveness of PPO in multiagent problems for discrete action space domains. Their model is optimised using a centralised reward and demonstrated to outperform baseline methods such as [83, 108].

Game-theoretic Credit Assignment for Multiagent RL. Several works have considered the Shapley value for credit assignment in multiagent RL [75, 143, 144, 149]. However, most aforementioned methods are devoted to discrete action spaces and are not directly applicable to continuous control. For example, [75] has shown state-of-the-art performance on Starcraft, but struggles when adapted to continuous control problems due to inaccurate estimations. Unlike most prior credit assignment methods, which only consider the Shapley value, we introduce semivalues to credit assignment in multiagent RL and define a more generic game-theoretic framework that encompasses a family of common solution concepts and analyze the relationship between them. Moreover, we propose a novel model-based multiagent RL framework for multiagent continuous control and show that our model-based RL module can better estimate the coalition values for credit assignment and improve the sample efficiency accordingly.

Model-based RL. Model-based RL approaches typically alternate between fitting a predictive model of the environment dynamics/rewards and updating the control policies. The model can be used in various ways, such as execution-time planning [26, 94], generating simulated experiences for training the control policy [61, 132]), etc. Our work is inspired by [37], which addresses the problem of error in long-horizon model dynamics prediction. In particular, [37] presents a hybrid algorithm that uses the model to simulate short-term horizon transition and rewards, while using Q-learning to estimate the long-term value beyond the simulation horizon. In the field of multiagent RL, [20] and [21] introduced Bayesian model-based approaches to multiagent RL for coalition formation and optimal exploration.

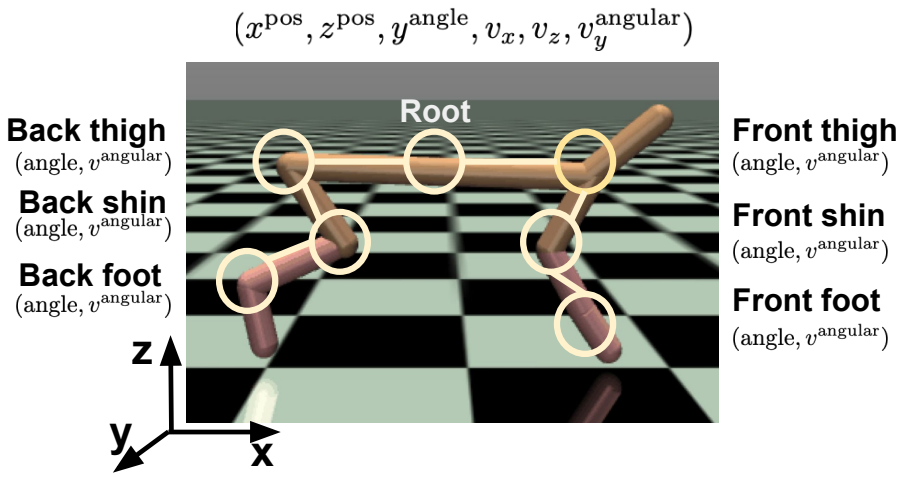


Figure 4.1: Multiagent MuJoCo Cheetah

In this chapter, we look at the fully-cooperative setting of robotic control. And we study model-based RL with centralised training, decentralised execution, which allows game-theoretic credit assignment for the multiagent continuous control problems.

4.3 Robot Joints as a Multiagent System

In this section, we introduce the kinematics tree of robots and how to model the robot joints as a multiagent system.

4.3.1 Kinematics Tree of Robots

A robot capable of performing continuous control can be defined via a kinematics tree [127, 145] of nested bodies, specifying the mechanical structure and physical properties of the robot. Within such a kinematics tree, there are three major types of entities: `Bodies, Joints and Actuators`. The `Bodies` represent the rigid bodies of the robot, e.g., foot. Physical properties such as the relative position and geometry are specified for each body. The `Joints` are moveable components which connect parent and child bodies, and create motion degrees of freedom between them. Typical joints include ball joints (3 rotational degrees of freedom), hinge joints (1 rotational degree of freedom) and slide joints (1 translational degree of freedom). For example, a cheetah robot (Figure 4.1) consists of a torso (root `body`) and the front/back legs, and each leg consists of three bodies (thigh, shin, and foot) which are connected by hinge joints, for example, the back foot (`body`) is attached to its parent back shin (`body`) through a hinge `joint`. Attached to the

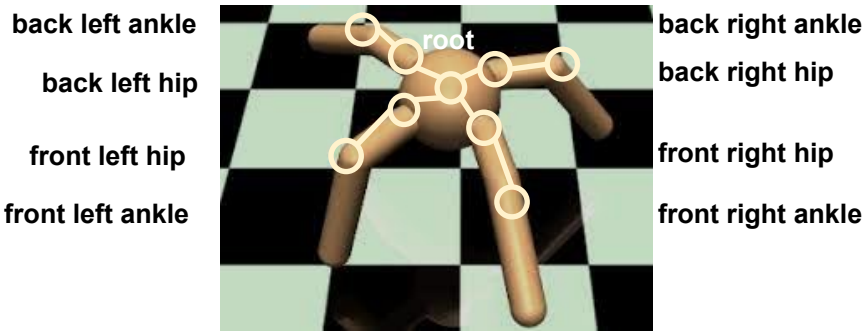


Figure 4.2: Multiagent MuJoCo Ant

joints are `Actuators` which are the components that actuate the joints. Together these joints allow the robot to perform translational and rotational movements.

Next, we introduce how to model the robot as a multiagent system. Figure 4.1 shows an example cheetah robot built using MuJoCo [13] (more details in Section 4.6.1). We model the robot as a multiagent system of N agents, where each agent represents: 1. a rigid body component (e.g., the back foot) of the robot, 2. the joint that attaches the body to its parent (e.g., the ankle joint which attaches the back foot to the back shin), and 3. any actuator attached to the joint. In this way, the agent receives input of the local sensor observations to its joint (e.g., positions, angles, translational and angular velocities, external forces and torques), and outputs a control action for the actuator.

4.3.2 Multiagent Control for Robot Locomotion

Typically in the robotic control RL settings, the actuators have a continuous action space which is normalised to $[-1, 1]$ for the convenience of learning. In this work, we focus on the robot locomotion task, which is a representative continuous robot control problem in this family. The objectives of robot locomotion tasks are to train the robot to learn control policies in order to transport from place to place through walking, hopping, swimming, etc.

Here we model the locomotion tasks as a fully-cooperative Multiagent decision making problem, described by a Markov game [79] denoted by a tuple $\langle N, \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle$, where N is the set of all agents $N = \{1, \ldots, n\}$, and $\mathcal{S}$ are global states of the environment. Each agent i has a continuous action space $\mathcal{A}_i$ and the combined action space of the robot is $\mathcal{A} = \mathcal{A}_1 \times \ldots \times \mathcal{A}_n$. At each step t, each agent i makes a local observation $s_t^i = o^i(s_t)$, and chooses an action simultaneously according to its own policy $a_t^i \sim \pi^i(s_t^i)$. Then, a combined action

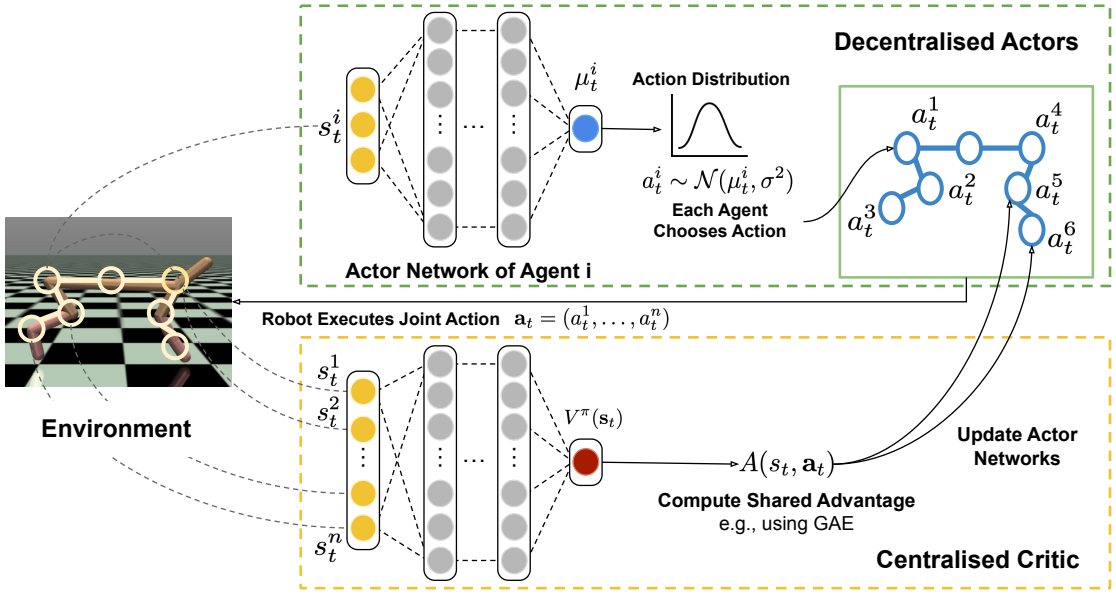


Figure 4.3: Multiagent PPO with Shared Advantage. The Multiagent PPO framework consists of two modules: (lower yellow box) the centralised critic module which estimates the centralised state-value function $V^{\pi}(s)$; and (upper green box) decentralised actors module with N agents, each updated with the shared advantage $A(s_t, \mathrm{a}_t)$ computed using the centralised state-value function (and additional information such as the n-step global returns) through methods such as generalised advantage estimation (GAE).

composed of the actions of all agents $\mathrm{a}_t = (a_t^1, \ldots, a_t^n)$ is performed on the environment (the combined policy of all agents is denoted as $\pi = (\pi^1, \ldots, \pi^n)$). Upon taking the action, all agents receive a shared reward $r_t = r(s_t, \mathrm{a}_t)$ which evaluates how well the robot performed (e.g., the distance travelled along a specified direction, the control energy consumption, the stability of the robot, etc), and the environment transitions to the next state according to the transition dynamics $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$. At each step, the agents aim to maximise the cumulative future discounted return $G_t = \sum_k \gamma^k r_{t+k}$ where $\gamma \in [0, 1]$ is a discount factor.

4.3.3 Multiagent PPO with Shared Advantage

Having formulated the multiagent decision problem, we next introduce an algorithm for optimising the agents' policies. One of the best known RL algorithms for continuous control is Proximal Policy Optimisation (PPO) [118], which is widely used due to simplicity in implementation, sample complexity, and ease of hyperparameter tuning. To learn the multiagent continuous control policies, we first extend the standard single-agent PPO algorithm to cooperative multiagent PPO and adopt the centralised training, decentralised execution (CTDE) paradigm [42][1]. Figure 4.3

[1]Figure 4.3 includes most of the necessary details on CTDE and PPO. For more details on CTDE please refer to Section 2.2.4.1, and more details on PPO and generalised advantage estimation (GAE) are in Section 2.2.3.1.

shows an illustration of the multiagent PPO framework which consists of two main modules, the *centralised critic* (yellow box) and *decentralised actors* (green box). In Table 4.1, we summarise the important notations of this chapter and where they are first introduced.

Decentralised Actors. Recall that an agent represents a robotic joint together with its actuator and attached body component. Associated to each agent i is one of the decentralised actors, which represents the policy π^i of the agent, and is often modelled by a neural network, with parameters denoted by θ. As shown in the figure, at each step t, the actor takes in the agent's local observation s_t^i, and outputs an action $a_t^i = \pi^i(s_t^i)$ for the associated actuator. Specifically, to account for the continuous action space, each agent's policy is modelled by a Gaussian distribution (with a static or learned variance parameter σ). At each step, the actor network outputs the mean μ_t^i of the Gaussian distribution, then the action is sampled from the Gaussian distribution $a_t^i \sim \mathcal{N}(\mu_t^i, \sigma)$. Together, the decentralised actors form a combined action $\mathrm{a}_t = (a_t^1, \ldots, a_t^n)$ which controls the robot to interact with the environment.

Centralised Critic. During training, the agents share a central critic module that evaluates the value of the global state $V(s_t)$. As introduced in Section 2.2.1, the state-value function $V(s_t)$ measures the expected discounted future return G_t from the state s_t onwards. As shown in the figure, the centralised critic is often modelled by a neural network, with parameters denoted by ω. Following the estimation of $V(s_t)$, the shared advantage function $A(s_t, \mathrm{a}_t)$ is computed using methods such as generalised advantage estimation (GAE), which measures how much better taking a_t in state s_t is compared with the policy's default behaviour.[2] The critic is only used during training to provide signals which guide the optimisation of the actors and will be dismissed in the execution phase—during the execution only the actors are used to choose the actions.

To optimise the centralised critic and decentralised actors, we follow the PPO algorithm: At each epoch k, the agents collect a set of trajectories $\mathcal{D}_k$ by operating the robot using their current policy $\pi_k = (\pi^1(\theta_k^1), \ldots \pi^n(\theta_k^n))$ in the environment, where θ_k^i is the neural network parameters of agent i in epoch k. Then we update the critic and actor parameters. The centralised critic neural network parameters ω are updated by minimising the mean squared error between the predicted

[2] for more details on GAE please refer to Section 2.2.3.

Table 4.1: Notations and references to their background sections for Chapter 4

i, t, k	Indices for agents, timesteps, iterations
π^i	Policy of agent i (Sec. 2.2, 4.3.3)
G_t	Discounted return from t onwards (Sec. 2.2)
$V^\pi(s), Q^\pi(s, \mathrm{a})$	State and action-value functions (Sec. 2.2.1.2)
$A^\pi(s, \mathrm{a})$	Advantage function (Sec. 2.2.3.1, 4.3.3)
$\omega, \theta, \phi_s, \phi_r$	parameters of V, π, f_s, f_r (Sec. 4.3.3, 4.5)
$v^C(s, \mathrm{a})$	Characteristic value of coalition (Sec. 4.4.1)
$\tilde{\mathrm{a}}_t$	Actions a_t masked by coalitions (Sec. 4.4.1)
$\mathcal{MC}^i(C, s, \mathrm{a})$	Marginal contribution of agent i to coalition C
$\varphi^i(v), p_c$	semivalues and its probability indices (Sec. 4.4.2)
$f_s(s, \mathrm{a}), f_r(s, \mathrm{a})$	dynamics and reward models (Sec. 4.5.1)
$\hat{s}_{t+1}, \hat{r}_t$	States/Rewards predicted by model (Sec. 4.5.1)

state value $V(s_t)$ and the empirical returns G_t from t onwards, computed from the collected data:

$$\omega \leftarrow \arg\min_{\omega} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \big(V_\omega(s_t) - \hat{G}_t\big)^2.$$

The actor of each agent is updated by optimising the PPO clipped surrogate objective [118]:

$$\theta^i_{k+1} = \arg\max_{\theta^i} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min\Big(\frac{\pi_{\theta^i}(a^i_t|s^i_t)}{\pi_{\theta^i_k}(a^i_t|s^i_t)} A(s_t, \mathrm{a_t}), \mathrm{clip}\big(\frac{\pi_{\theta^i}(a^i_t|s^i_t)}{\pi_{\theta^i_k}(a^i_t|s^i_t)}, 1-\epsilon, 1+\epsilon\big) A(s_t, \mathrm{a_t})\Big),$$

where $A(s_t, \mathrm{a}_t)$ is the shared advantage function computed using the critic through GAE. Intuitively, if the advantage $A(s, a)$ is positive, the actor parameters update in the direction where this action becomes more likely. PPO avoids parameter updates that change the policy too much by clipping the ratio between old and new policies to be within $[1 - \epsilon, 1 + \epsilon]$.

4.4 Multiagent PPO with Credit Assignments using Semivalues

In the above multiagent PPO algorithm, the shared advantage $A(s_t, \mathrm{a}_t)$ only evaluates the quality of the combined action of the robot, which does not specify each agent's individual contribution. As we will discuss in the experiments (Section 4.6.2.1), failing to address per-agent contribution can result in low sample efficiency. For example, an under-actuated agent can equally share and update its policy using the global advantage as an agent which made significant contributions. To deal with these issues, we leverage the credit assignment methods from cooperative game theory, and present a generic game-theoretic framework for agent-specific advantage computation in Multiagent PPO.

4.4.1 The Characteristic Function

To compute the value of an agent assigned by the game-theoretic solution concepts, we first need to define the characteristic function. Compared with Chapter 3 where the payoff allocation is performed post-hoc, a major difference here is that the credit assignment is performed at each timestep hence the characteristic function needs to take into account the state of the robot and the executed action: At each timestep t, given the current state s_t and agents' joint action $\mathrm{a}_t = (a_t^1, \ldots, a_t^n)$, let the value of a coalition C be defined as:

$$v^C(s_t, \mathrm{a}_t) = Q^\pi(s_t, \tilde{a}_t^1, \ldots, \tilde{a}_t^n), \text{ where} \tag{4.1}$$
$$\tilde{a}_t^i = \begin{cases} a_t^i & \text{if } i \in C \\ a_{\text{default}} & \text{if } i \in N \setminus C, \end{cases}$$

where $Q^\pi(s_t, \tilde{\mathrm{a}}_t)$ is the action-value which measures the expected future return of the robot when starting from s_t and taking action $\tilde{\mathrm{a}}_t$ (first defined in Equation (2.5) in Section 2.2.1). Inside the action value function, $\tilde{a}_t^i$ denotes the action taken by agent i is replaced by a default one (e.g., a zero action) if i is outside the coalition, which is a widely adopted practice introduced by [147]. For example, recall that the control action of each agent corresponds to the torque exerted on the joint by the actuator and is typically normalized to the real interval $[-1, +1]$ for the convenience of learning. In this case, the agents playing default zero actions correspond to joints that are un-actuated during simulation or real-world execution, which is an intuitive model for the agents not contributing. An illustrative real-world example would be a legged robot with an un-actuated ankle, and in such a case, alternative approaches such as physically removing the agents (joints) would be practically impossible, as this would damage the robot and disable its movement. In addition, to follow the game-theoretic conventions where empty coalitions have zero value, we can normalise the characteristic value function by subtracting a baseline value $Q^\pi(s_t, \mathrm{a}_{\text{default}})$. Nevertheless, the marginal contributions will stay invariant so we will use the above definition for simplicity.

Given this characteristic function, agent i's marginal contributions towards coalition C represent the difference made by i's chosen action towards coalition C, compared with the default action, that is,

$$\mathcal{MC}^i(C, s_t, \mathrm{a}_t) = v^{C \cup \{i\}}(s_t, \mathrm{a}_t) - v^C(s_t, \mathrm{a}_t), \tag{4.2}$$

4.4.2 Agent-Specific Advantage

We now introduce a generic game-theoretic credit assignment framework for multiagent continuous control using semivalues. As we discussed, the semivalues are a wide class of solution concepts that encompass many common credit assignment methods such as the Shapley value, the Banzhaf value, etc. Given the set of agents N and their marginal contributions, the semivalue of an agent i is defined as follows,

$$\varphi^i(v) = \sum_{C \subseteq N \setminus \{i\}} w_{c,N} \mathcal{MC}^i(C) \text{ where } c = |C|, \sum_{c=0}^{|N|-1} w_{c,N} \binom{|N|-1}{c} = 1.$$

As we have seen in the previous chapter, the semivalue of an agent can also be defined as a weighted sum over its *average marginal contributions* by coalition sizes c. We can rewrite the semivalues as:

$$\varphi^i(v) = \sum_{c=0}^{|N|-1} \alpha_c \frac{\sum_{C \subseteq N \setminus \{i\}, |C|=c} \mathcal{MC}^i(C)}{\binom{|N|-1}{c}}, \text{where } \sum_{c=0}^{|N|-1} \alpha_c = 1, \tag{4.3}$$

where the denominator $\binom{|N|-1}{c}$ is the number of size-c coalitions excluding player i. In this chapter, we will use the superscript to denote the players i and coalitions C, and the subscripts denote timesteps t, as shown in Table 4.1. Recall that the semivalue is a weighted sum of an agent's average marginal contributions towards different sized coalitions, where the importance weights α_c form a probability distribution. In particular, the Shapley value places uniform weights on all coalition sizes: $\alpha_c = \frac{1}{|N|}$; and the Banzhaf value has a bell-shaped distribution, i.e., $\alpha_c = \frac{1}{2^{|N|-1}} \binom{|N|-1}{c}$. Any other probability distribution α_c also specifies a semivalue.

At each timestep, the semivalue φ^i of an agent computes the contributions of the agent's chosen action towards the global future return of the robot and hence will be used as the agent-specific pseudo advantage for multiagent PPO updates. We call the semivalues "pseudo" advantage because they are not defined strictly as $A(s,a) = Q(s,a) - V(s)$. Nevertheless, the semivalues provide an effective proxy of advantage as they can evaluate the individual contribution of an agent's action towards the global value. Since RL training procedures typically require millions of timesteps, enumerating all possible coalitions and computing their characteristic value at each time step is inefficient. A simple procedure for estimating the semivalue is through Monte-Carlo sampling and output the average over the agent's marginal contribution towards all sampled coalitions. For each sample drawn, we can first sample a coalition size c_m according to the semivalue distribution p_c, then uniformly sample a coalition C_m of the size c_m.

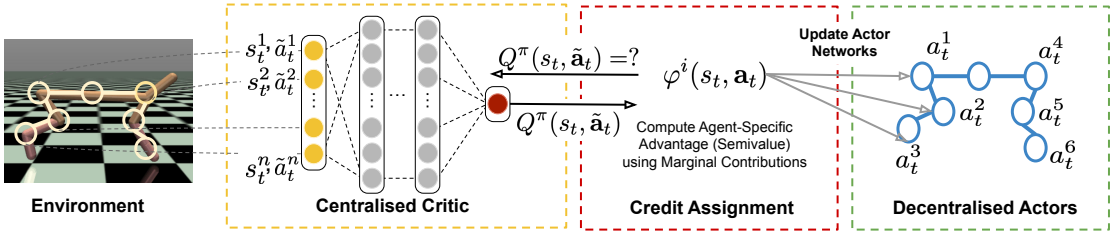


Figure 4.4: Agent-Specific Advantage for Multiagent PPO. The (model-free) credit assignment framework consists of three modules: (yellow) the centralised critic module which estimates the centralised action-value function $Q^\pi(s_t, \tilde{\mathrm{a}}_t)$; (red) the credit assignment module which computes the agent-specific advantage $\varphi^i(s_t, \tilde{\mathrm{a}}_t)$) using the action values provided by the critic; and (green) decentralised actors module with N agents, each updated using the agent-specific advantage computed from the credit assignment module.

4.5 Model-based Advantage Estimation

The semivalues (e.g., the Shapley and Banzhaf values) as agent-specific pseudo advantages allow us to evaluate the contribution of each agent and its chosen action. *However, how can we obtain the coalitions' values, when only the value of the grand coalition is available (i.e., the global reward and returns of the whole system)?*

An obvious approach is to perform extra simulations, where for each simulation the grand coalition is replaced by a different coalition of agents. This, however, would require exponentially more simulations which is expensive for multiagent RL domains in general and especially for robotics. Another (model-free) approach which does not require extra simulations is to infer the value of coalitions $v^C(s_t, \tilde{\mathrm{a}}_t) = Q^\pi(s_t, \tilde{\mathrm{a}}_t)$ based on the present simulations. A possible implementation of this approach for multiagent PPO is shown in Figure 4.4. To compute the agent-specific advantage, the credit assignment module queries the different coalition values from the centralised critic network and compute the semivalues according to the marginal contributions of the agents towards coalitions of other agents. The critic network predicts the state-action values of the pair $(s_t, \tilde{\mathrm{a}}_t)$ where $\tilde{\mathrm{a}}_t$ is the masked action according to the coalition C. However, this approach can suffer from inaccurate coalition value estimations in the multiagent robotics control domain due to the continuous, infinite action space.

To accurately estimate the value of different coalitions, we draw inspiration from model-based RL and propose to incorporate an additional `model` of the environment transition dynamics and rewards. In this way, we can obtain a better coalition value estimation through model-based simulations of the different coalitions. Meanwhile, optimisation of the model is straightforward

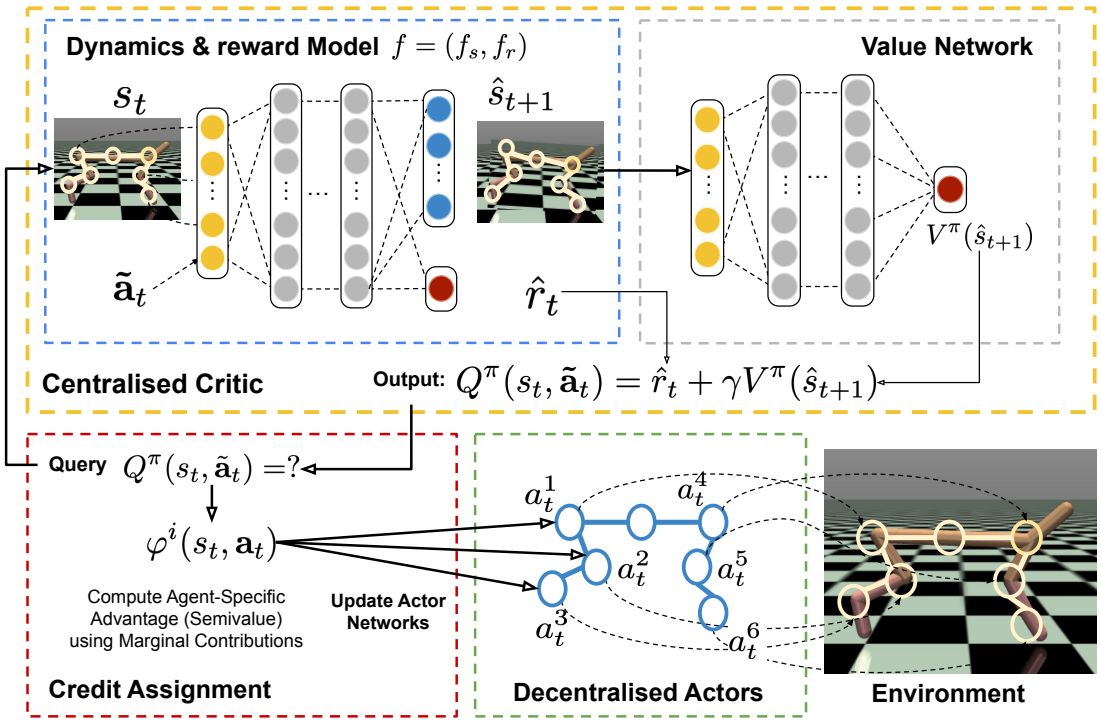


Figure 4.5: Model-based Credit Assignment for Multiagent PPO. Our framework consists of three modules: 1. (yellow) the centralised critic module which consists of a dynamics & reward model $f = (f_s, f_r)$ and a centralised state-value function $V^\pi(s)$; 2. (red) credit assignment module which computes the semivalue based agent-specific advantage values $\varphi^i(s_t, \mathrm{a}_t)$; 3. (green) decentralised actors which are policies of the agents, and are updated using the agent-specific advantage from the credit assignment module. To compute the agent-specific advantage, the credit assignment module queries the coalition values $v^C(s_t, \mathrm{a}_t) = Q^\pi(s_t, \tilde{a}_t)$ from the critic, then compute the semivalue of an agent using its marginal contributions. Inside the critic, to compute $Q^\pi(s_t, \tilde{a}_t)$, the model first predicts the next state and immediate reward $(\hat{s}_{t+1}, \hat{r}_t)$, then pass them to the value network, which evaluates $V^\pi(\hat{s}_{t+1})$. Then, the centralised critic computes and returns the queried coalition value Q^π to the credit assignment module.

(via supervised learning) and does not require extra environment interactions. We will introduce the dynamics and reward model in the next section.

4.5.1 Dynamics and Reward Model

As shown in Figure 4.5 (top left blue module), we use a *model* $f = (f_s, f_r)$ of the environment, which is often modelled by neural networks with parameters $\phi = (\phi_s, \phi_r)$. The model consists of two parts – a transition dynamics model f_s which maps the state-action pair to the next state, and a reward model f_r which maps the state-action pair to the immediate reward. In particular, we adopt a formulation of the dynamics model where f_s maps the state-action pair to the difference between the next and current state. The dynamics and reward models are shown

in the following two equations, respectively,

$$\forall s_t \in \mathcal{S}, a_t \in \mathcal{A}, \quad \hat{s}_{t+1} = f_s(s_t, a_t) + s_t, \tag{4.4}$$

$$\hat{r}_t = f_r(s_t, a_t), \tag{4.5}$$

where hat $\hat{}$ denotes predicted values as apposed to the observed values. Both models can be learned from scratch during the RL procedure through supervised learning. To optimise the model, we can perform regression and minimize the mean-squared error between the predicted next states/rewards and actual observed ones, that is,

$$\min_{\phi_s} \frac{1}{|\mathcal{D}|} \sum_{\langle s_t, s_{t+1}, \mathrm{a}_t, r_{t+1} \rangle \in \mathcal{D}} \|s_{t+1} - (s_t + f_s(s_t, \mathrm{a}_t))\|^2 \tag{4.6}$$

$$\min_{\phi_r} \frac{1}{|\mathcal{D}|} \sum_{\langle s_t, s_{t+1}, \mathrm{a}_t, r_{t+1} \rangle \in \mathcal{D}} \|(r_t - f_r(s_t, \mathrm{a}_t)\|^2. \tag{4.7}$$

The models are part of the centralised critic which is only used in training, hence only one centralised model is needed and can compensate for the global state of the multiagent system. During each epoch of training, we alternate between model fitting and the PPO agent updates, using the experiences collected from interactions with the environments. Moreover, inspired by model-based value expansion [37], we only use the model for generating imaginary simulations of short-horizons (e.g., one step), which mitigates the error-prone model predictions due to drifting in long-horizons.

4.5.2 Estimating Coalition Values using the Model

We now show how to apply the models to coalition value estimations by performing imaginary simulations of the coalitions. This is achieved by first performing predictions of the one-step transition dynamics and reward, then evaluating the coalition values using the centralised value function. And an illustration of the model-based coalition value estimation procedure is shown in the centralised critic module (yellow box) in Figure 4.5. Specifically, given the state, action pair (s_t, a_t) and a specific coalition C, we first mask the action by the coalition $\tilde{\mathrm{a}}_t$ according to Equation (4.1), and use the model to predict the next state $\hat{s}_{t+1}$ and reward $\hat{r}_t$. Then, we can estimate the state-value $V^\pi(\hat{s}_{t+1})$ of the predicted next state using the centralised value network.

Finally, we can compute the coalition value (i.e., the action-value $Q^\pi(s_t, \tilde{\mathrm{a}}_t)$ for the current state and masked action) through the Bellman equation, as shown in the following:

$$\begin{aligned} Q^\pi(s_t, \tilde{\mathrm{a}}_t) &= \mathbb{E}[r_t + \gamma V^\pi(s_{t+1}) |\ s_t, a_t] \\ &= \mathbb{E}[\hat{r}_t + \gamma V^\pi(\hat{s}_{t+1}) |\ s_t, a_t] \\ &\approx f_r(s_t, \tilde{\mathrm{a}}_t) + \gamma V^\pi(s_t + f_s(s_t, \tilde{\mathrm{a}}_t)). \end{aligned} \tag{4.8}$$

This completes our model-based credit assignment framework for multiagent PPO. To summarise, the framework consists of three main components, the decentralised actors, the centralised critic, and the credit assignment module. Furthermore, the centralised critic consists of two sub-modules, i.e., the dynamics and reward model, and the centralised value network. An illustration of the complete framework is shown in Figure 4.5, and the detailed training procedure is specified in Algorithm 7. For each iteration of training, we collect experiences by running the robot in the environment using the decentralised actors, and use the collected experiences to fit the dynamics and rewards model. Then, to produce the agent-specific advantage for each agent i for each timestep, the credit assignment module first samples coalitions $C_m \subseteq N \setminus \{i\}$, and computes the marginal contributions of i towards the sampled coalitions $\mathcal{MC}_i(C_m) = v(C_m \cup \{i\}) - v(C_m)$. To obtain the value of a coalition C, that is, $v^C(s_t, \tilde{\mathrm{a}}_t) = Q^\pi(s_t, \tilde{\mathrm{a}}_t)$, the credit assignment module queries the centralised critic module (yellow in Figure 4.5) – In the critic, the model (blue) first simulates and predicts the next state $\hat{s}_{t+1} = f_s(s_t, \tilde{\mathrm{a}}_t)$ and reward $\hat{r}_t = f_r(s_t, \tilde{\mathrm{a}}_t)$, then the next state $\hat{s}_{t+1}$ is sent to the centralised value network (grey), which estimates $V^\pi(\hat{s}_{t+1})$. Together with the reward $\hat{r}_t$ predicted by the reward model, the critic module returns the value of the coalition C_m to the credit assignment module as $Q^\pi(s_t, \tilde{\mathrm{a}}_t) = \hat{r}_t + \gamma V^\pi(\hat{s}_{t+1})$. Finally, having computed the agent-specific advantage for each agent, we can update the decentralised actor networks (green) and centralised value networks (grey) through multiagent PPO.

4.6 Experiments

In this section, we present empirical results of our framework model-based credit assignment with multiagent PPO on continuous robotic control tasks.

Algorithm 7 Model-based Multiagent Credit Assignment

Input: $N = \{1, \dots, n\}$: the set of all agents.
Initialise: model $f = (f_s, f_r)$, centralised critic V_ω, decentralised actors π^i_θ for each agent $i \in N$
for iterations $k = 0, 1, 2, \dots$ **do**
    Collect sets of trajectories $\mathcal{D}_k = \{\tau\}$ by running policy $\pi_k = (\pi^1_{\theta^1_k}, \dots \pi^n_{\theta^n_k})$ in the environment. Compute returns $\hat{G}_t$
    # Fit the dynamics/reward model (in minibatches):
        $f_s \leftarrow \arg\min_{f_s} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \|s_{t+1} - (s_t + f_s(s_t, \mathrm{a}_t))\|^2$
        $f_r \leftarrow \arg\min_{f_r} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \|(r_t - f_r(s_t, \mathrm{a}_t)\|^2$
    # Compute per-agent advantage φ_i for all timesteps t:
        Sample coalitions C_m
        Compute coalition values by model f and critic V_{ω_k}:
            $v(C_m, s_t, \mathrm{a}_t)) = f_r(s_t, \tilde{\mathrm{a}}_t) + \gamma V^\pi_\omega(f_s(s_t, \tilde{\mathrm{a}}_t) + s_t)$
        Compute marginal contribution $\mathcal{MC}^i(C_m, s_t, \mathrm{a}_t)$
        Compute semivalues φ^i using the marginal contributions.
    # For each agent i, update the policy by maximising the PPO-Clip objective:
        $\theta^i_{k+1} = \arg\max_{\theta^i} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min\Big(\frac{\pi_{\theta^i}(a^i_t|s^i_t)}{\pi_{\theta^i_k}(a^i_t|s^i_t)} \varphi^i(s_t, \mathrm{a_t}),$
            $\mathrm{clip}\big(\frac{\pi_{\theta^i}(a^i_t|s^i_t)}{\pi_{\theta^i_k}(a^i_t|s^i_t)}, 1-\epsilon, 1+\epsilon\big)\varphi^i(s_t, \mathrm{a_t})\Big),$
    # Fit the centralised critic by regression:
        $\omega_{k+1} = \arg\min_\omega \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \big(V_\omega(s_t) - \hat{G}_t\big)^2$
end for

4.6.1 Experiment Setups

For the experiments, we use OpenAI Gym MuJoCo [13, 140] locomotion tasks, which are the standard benchmarks for continuous control with RL [145, 94]. Two exemplar robot locomotion tasks are selected as our case studies: (1) the Cheetah movement in 2D space and (2) the Ant movement in 3D space. The cheetah model in MuJoCo has 7 joints, including one root joint and 6 actuated joints each paired with one actuator. Following the definition in Section 4.3, we consider each joint as an agent with an actuator that applies a joint torque. For all agents, the action space is normalised to $[-1, 1]$. The root agent observes the position, angle, velocity and angular velocity and for other agents, their observation includes the relative angle with respect to the body that they are attached to, and the angular velocity. The global observation utilised by the centralised value network and the dynamics/rewards model (c.f. Figure 4.5) is given by a concatenation of local observations. The objective of the agent is to move forward in the x-direction. On the other hand, the Ant model can move in a 3D space and the agents are similarly defined as in the Cheetah.

Notably, the agents of an Ant model in MuJoCo can additionally observe external forces such as friction. The objective of the Ant locomotion task is also to move forward in the x-direction.

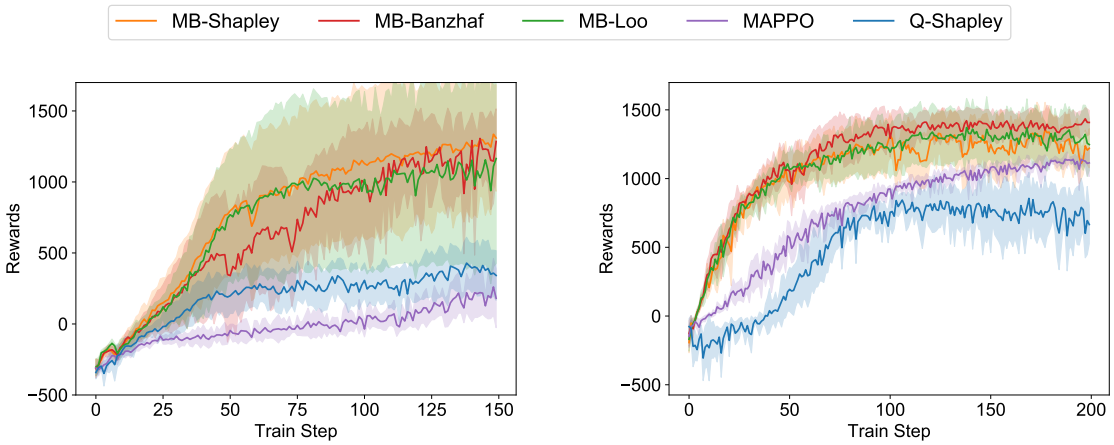
4.6.1.1 Models and Training Details

Our multiagent system consists of a model $f = (f_s, f_r)$ of the environment, the centralised critic networks, and the decentralised actors. We use PyTorch [104] for implementing and optimising the neural network models. For the dynamics model f_s, we use 4 fully-connected layers with ReLU activation function [99] and a linear output layer with the dimension of 128. For the reward model f_r, we use 3 fully-connected layers with ReLU activation function and a linear output layer with the dimension of 128. The model is optimised in minibatches of size 64 using Adam optimiser [64] with the learning rate of 10^{-3}. For the centralised critic, we use 3 fully-connected layers of the dimension of 32 with Tanh [99] activation and a linear output layer which produces the value of the input state. For the decentralised actors, each actor uses 3 fully-connected layers of the dimension of 32 with Tanh activation and an output layer with Tanh activation. The default actions a_{default} are given by zero vectors. Both the critic and the actor parameters are optimised using the Adam optimiser. The learning rate for the critic is 10^{-3} and is 3×10^{-4} for the actor. All graphs plot mean and standard deviation across 5 seeds. The reward function of the MuJoCo locomotion tasks include the robot's velocity in the x direction and a penalty on the action size, which is provided by OpenAI Gym [13]. And the implementation details including most hyperparameter choices are from the code provided by the single-agent PPO from Stable-baselines [58], a widely used package for RL algorithms.

4.6.2 Overall Results

Figure 4.6 shows the overall performance of our algorithms and baseline algorithms on the MuJoCo locomotion tasks with cheetah and ant robots. The y-axis shows the episode rewards and x-axis shows the train steps. In particular, we follow Equation 4.3 to implement the following variants of our semivalue based credit assignment algorithms. `MB` is for Model-based coalition value estimation:

- `MB-Shapley`: Model-based credit assignment with the Shapley value ($\alpha_c = \frac{1}{|N|}$);
- `MB-Banzhaf`: Model-based credit assignment with Banzhaf value ($\alpha_c = \frac{1}{2^{|N|-1}}\binom{|N|-1}{c}$);



(a) Cheetah (Rewards)

(b) Ant (Rewards)

Figure 4.6: Average Rewards Multiagent Cheetah and Ant

- `MB-Loo`: Model-based credit assignment with the Leave-one-out value ($\alpha_c = \mathbb{1}_{c=|N|-1}$), where we compute the marginal contribution of agents towards all other agents;

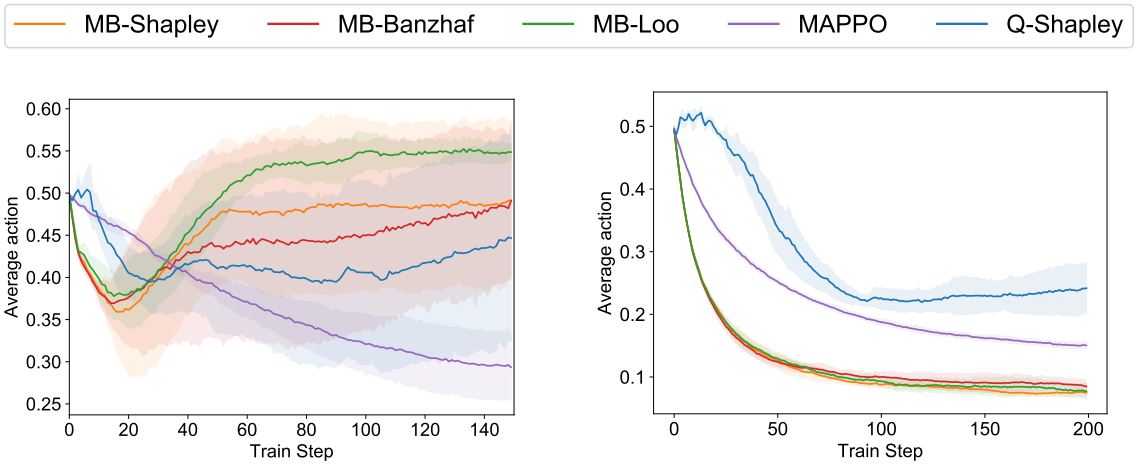
For *baselines*, we compare with the following state-of-the-art algorithms in multiagent RL that use the Shapley value-based credit assignment and the global advantage respectively:

- `Q-Shapley` [75]: we adapt the discrete domain algorithm which used Shapley value for credit assignment, to our multiagent PPO which can be used for continuous control. The coalition values are computed using a centralised critic of Q-values in a model-free manner.

- `MAPPO` [150]: In `MAPPO` (Multiagent PPO), the centralised critic computes a shared advantage function for all agents based on the global reward signal. And the agents are trained using centralised training, decentralised execution.

4.6.2.1 Shared Advantage vs. Agent-specific Advantage.

We are now in a position to present the results and findings. In both the cheetah (Figure 4.6a) and ant (Figure 4.6b) cases, we observe that our credit assignment methods that use model-based estimation (i.e., `MB-Shapley`, `MB-Banzhaf`, `MB-Loo`) significantly outperform multiagent PPO (`MAPPO`) in terms of both the *average rewards* and *sample efficiency*. For example, in the Ant case, our model-based algorithm with Banzhaf credit assignment (`MB-Banzhaf`) quickly reaches the reward of 1000 within 50 train steps, while `MAPPO` needs around 150 train steps to reach the same level. `MB-Banzhaf` also converges to a higher average reward ($\sim$ 1400) than



(a) Cheetah (Average Actions)

(b) Ant (Average Actions)

Figure 4.7: Average Actions Multiagent Cheetah and Ant

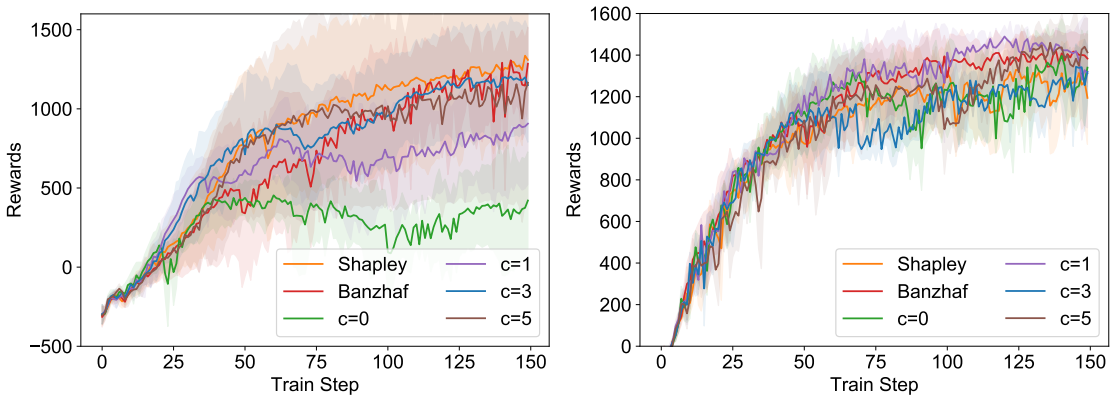
`MAPPO` ($\sim$ 1100). In all our results, performing credit assignment and evaluating the per-agent contribution is consistently better than agents using a shared advantage function.

4.6.2.2 Model-based vs. Model-free Credit Assignment.

We next compare our `MB-Shapley` with the baseline method `Q-Shapley` [75] in order to understand the role of model-based credit assignment. In our `MB-Shapley`, the coalition values are estimated using the dynamics/reward model and the state-value critic V^{π}. While in `Q-Shapley`, the coalition values are estimated in a model-free way with action-value critic Q^{π}. As shown in Figure 4.6a and 4.6b, for both Cheetah and Ant locomotion tasks, `MB-Shapley` outperforms `Q-Shapley`. This shows that when only the reward of the grand coalition (i.e., the whole robot) is provided by the simulations, it is difficult to infer the values v^{C} of different coalitions using a model-free Q-value critic. By contrast, a better estimation of values of different coalitions is obtained with the predictions of our model-based approach, which enables more effective agent-specific credit assignment and sample-efficient learning.

4.6.2.3 Robustness of Different Semivalue Variants.

Our model-based coalition value estimation supports the broad class of semivalue credit assignments. Still, from Figure 4.6, we observe that the agents trained using different selected semivalues (`MB-Shapley`, `MB-Banzhaf`, `MB-Loo`) ($\sim$ 1200 for Cheetah and $\sim$ 1300 for Ant) all outperform the other baseline methods. In the case of cheetah (Figure 4.6a), both



(a) Cheetah **(b)** Ant

Figure 4.8: Results for Varying Coalition Sizes

`MB-Shapley` and `MB-Banzhaf` yield the highest rewards, and in the case of ant (Figure 4.6b), `MB-Banzhaf` yields the highest rewards. Overall, `MB-Banzhaf` has the lowest variance among the three variants.

4.6.2.4 Average Actions

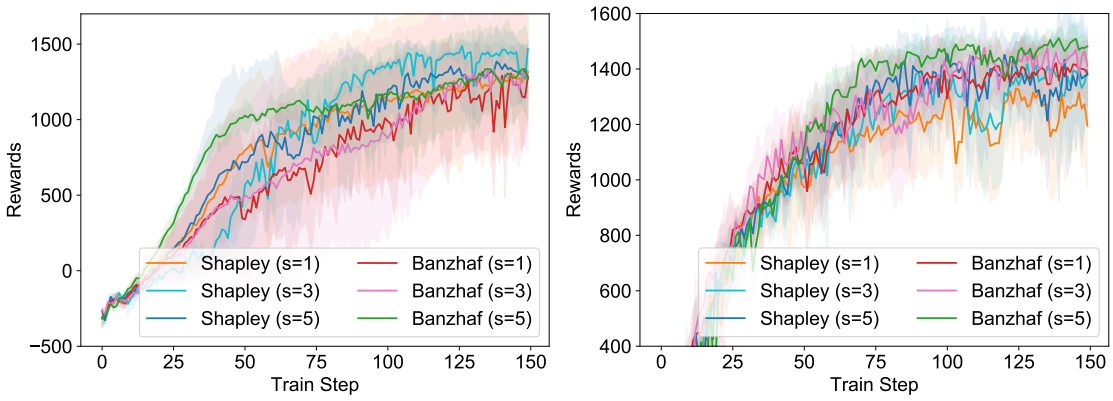
Figures 4.7a and 4.7b show the mean absolute value of actions, averaged across the agents (joints actuators). This quantity refers to the controller output (normalised between [0, 1]), and higher action values mean higher energy consumption in a practical context. In cheetah, `MB-Loo` has the highest average action, with `MB-Shapley` and `MB-Banzhaf` having medium values. In the ant case, `MB-Shapley`, `MB-Banzhaf`, and `MB-Loo` all have low control output. We observe that the ant robot trained using `MB-Shapley`, `MB-Banzhaf` learns to move forward using a subset of joints. By contrast, other joints are mainly learned for steering, implying that using these two semivalues can also encourage diverse roles of different agents in a locomotion task.

4.6.3 Component-wise Analysis

Apart from the main results, we provide further analysis on the changing hyper-parameters, i.e., variations of the coalition sizes and sample sizes.

4.6.3.1 Variation in Coalition Sizes

In this section, we discuss the model-based semivalues from the perspective of coalition sizes. Figures 4.8a and 4.8b show the performance of agents learned using two semivalues `MB-Shapley`



(a) Cheetah **(b)** Ant

Figure 4.9: Results for Varying Sample Sizes

($\alpha_c = \frac{1}{N}$), `MB-Banzhaf` ($\alpha_c = \frac{1}{2^N}\binom{N-1}{c}$), and some semivalues defined by $\alpha_c = \mathbb{1}_{c=x}$, e.g., a semivalue with $\alpha_c = \mathbb{1}_{c=3}$ is an agent's average marginal contributions towards size-3 coalitions. By comparing the performances of the semivalues of single coalition sizes, we can draw useful insights into the cooperation mode among the joints of the robot. In the case of cheetah (Figure 4.8a), the semivalues that use small coalition sizes (e.g., $\mathbb{1}_{c=0}$) yield unsatisfactory performance, while those using mid-sized coalitions (e.g., $\mathbb{1}_{c=3}$) yield the highest reward among the semivalues defined by a single coalition size, which may be due to the fact that there are three joints for both the cheetah's front and back leg. In the case of the ant (Figure 4.8b), the semivalues focused on coalitions of heterogeneous sizes all yield high rewards. This suggests that the agents' actions in the cheetah case are more dependent on each other, hence they require agents to closely coordinate their actions. Whereas for the ant, the agents have more distinct roles and their actions are more independent hence even forming smaller coalitions would suffice. By using the Shapley value and Banzhaf value, we take into account all coalition sizes (i.e., the Shapley value uniformly weighs all coalition sizes while Banzhaf value puts higher weights on mid-sized coalitions). Compared with the semivalues which use single coalition sizes, this effectively helps to reduce the hand-engineering for choosing the coalition size and still yield robust performance of the robot across different environments. This is also demonstrated by the performance of both the Shapley and Banzhaf value variants compared with the best-performing semivalues of single coalition sizes.

4.6.3.2 Variation in Sample Sizes

Lastly, we examine how the number of coalitions sampled per agent per step (when estimating the semivalues) affects the robot's performance. We compare `MB-Shapley` for the number of samples of $s = 1, 3$, and 5. We do the same for `MB-Banzhaf`. Figures 4.9a and 4.9b show that for both robots and both semivalue variants, reducing the number of sampled coalitions per node does not drastically decrease the performance in terms of reward and sample efficiency. Even using a small number of sampled coalitions, such as one per agent in each step, can still yield sufficiently good performance. We conjecture that this robustness property arises because, despite the small number of sampled coalitions per step, an RL algorithm typically runs on the scale of million timesteps. (Note that timestep is different from the train step, i.e, iteration in our graphs, which correspond to 10 training epochs each for 2048 rollout steps). Hence, the number of sampled coalitions throughout the entire learning phase often will be sufficiently large. Importantly, because RL requires many credit assignment computations where each involves neural network forward propagation, this property helps us significantly reduce computation cost without compromising the learning performance.

4.7 Conclusions and Future Work

In this chapter, we studied multiagent continuous control for robotics with model-based game-theoretic credit assignments. We first modelled robot joints as a fully-cooperative multiagent system and optimised the system using a multiagent version of PPO. We then proposed a generic game-theoretic credit assignment framework using semivalues for evaluating agent-specific advantage functions. Furthermore, we proposed a model-based framework that significantly improved estimations of coalition values, which empowers the application of game-theoretic credit assignments for multiagent continuous control tasks. Finally, we empirically demonstrate that our model-based credit assignment leads to sample-efficient and robust multiagent learning on MuJoCo robot locomotion tasks.

Future work: An interesting future direction is to study other continuous control algorithms. Moreover, we chose to study the MuJoCo locomotion tasks as the robot joints naturally form a multiagent system, and the MuJoCo platform provide a open-source, accessible simulation platform for continuous control in robotics. Therefore, it is one of the most widely adopted

benchmark for comparing continuous RL algorithm performance. In the future, it would be interesting to apply our method to other robot variants and real-world multiagent robotic applications such as multi-arm manipulation [106], multi-robot manipulation with legged robots [93] and formation control of UAV teams [101].

Moreover, it would be interesting to study the model-based RL methods with other configurations such as longer prediction horizons. Though we demonstrated the usefulness of semivalues as a reward signal for guiding the learning process, the semivalues can still be used for post-hoc evaluation of the agents in the multiagent RL systems. This may help to interpret and understand the roles and contributions of each team member. Finally, in this chapter, we focus on a single robot as a multiagent system, and we believe that multi-robot systems such as robotic rescue teams, exploration robot teams can also make use of our proposed model-based credit assignment multiagent PPO framework.

5
Conclusions and Future Perspectives

5.1 General Conclusions

In this thesis, we identified and investigated the challenges when applying game-theoretic solution concepts for payoff allocation in multiagent machine learning systems. Two major directions are explored: evaluating agents' contributions in submodular machine learning systems, and credit assignment for multiagent reinforcement learning systems. The former studies game-theoretic payoff allocation methods as a post-hoc analysis of agents' contributions and malicious replication manipulations. The latter looks at co-evolving agents and integrates the game-theoretic credit-assignment into the training loop as a reward signal.

In the first direction, we studied the behaviours of semivalues in submodular machine learning systems. We started by looking at a classic submodular facility location problem and derived closed-form solutions for the Banzhaf and Shapley value. We then showed that under the submodularity assumption, the agents' average marginal contributions decrease monotonically with respect to coalition sizes. This finding allows us to compare groups of coalitions with different sizes, rather than comparing coalitions only with set inclusion relations. We then studied the behaviours of semivalues against replication manipulation, where an agent tries to improve its own payoff through replicating its resources. The results are as follows: we found that the Shapley value is vulnerable against replication, that is, the total payoff to an agent according to the Shapley value monotonic increases and converges to its characteristic value. On the other hand, the Banzhaf value is robust against replication, that is, the total payoff of the malicious agent is neutral in the first replication, then monotonic decreases in subsequent replications. Finally, we showed a general condition that governs the replication-robustness of semivalues.

Following the theoretical study, we applied our results for data valuation in an emerging real-life application, i.e., the machine learning data market. We formulated the data market as a submodular game and studied the data replication attack, where a data seller replicates its data to gain a higher payoff. We then studied some related attack models, where the agents perform more sophisticated manipulations to avoid being easily identified, such as data perturbation, splitting, or subset replication. To address the expensive computation cost for valuating coalitions in machine learning systems, we studied efficient sampling methods for estimating the semivalues. Finally, we empirically validated our results on the facility location function and real-life machine learning datasets.

In the second direction, we investigated a rapidly emerging application of multiagent reinforcement learning for robotic control. Different from the previous application where payoff allocation was used for post-hoc analysis, the payoff allocation in multiagent reinforcement learning is integrated into the training loop, providing crucial learning signals for the agents. Specifically, we modelled the robot as a multiagent system where agents correspond to the connected, moveable components such as joints. To allocate the credit for each agent, we formulated the multiagent interaction as a cooperative game and computed agent-specific advantage values using semivalues and multiagent PPO. During the computation, we identified the challenge of estimating the coalition values due to the high-dimensional, continuous state and action space. To address the challenge, we proposed the model-based multiagent credit assignment framework, which produces an improved estimation of the coalition values through simulated experiences from a model of the environment dynamics. To the best of our knowledge, this is the first application of model-based deep reinforcement learning methods to multiagent credit assignments.

5.2 Future Perspectives

While the thesis provides an important step towards the application of game-theoretic payoff allocation in multiagent machine learning systems, there remain many important open challenges and opportunities.

In the following, we will first discuss some future directions which follow from both of our studied applications, then discuss their independent future opportunities:

- One of the most important open challenges is the *efficient computation or estimation of game-theoretic solution concepts*. Efficient computation has been an important topic in cooperative game theory. The computation cost is even exacerbated in their machine learning applications. This is because unlike conventional cooperative games where the values of coalitions are often provided as an oracle, in multiagent machine learning systems, a pre-existing characteristic function is often absent. Consequently, computing the payoff allocation (exactly) requires evaluating all possible coalitions, each corresponds to training or inferencing of machine learning models which are time and resource consuming. As we have seen from our studied applications, the computational issue may come from many aspects such as: 1. the number of coalition valuations increase exponentially as a function

of the number of agents; 2. the payoff allocation needs to be computed per training step when used as a learning signal for multiagent RL, and 3. furthermore, due to environmental constraints, coalition values may not even be obtainable through the ML model evaluations and needs to be inferred. In this thesis, we addressed these problems through sampling-based methods and an additional model of the environment. However, efficient computation has been an ongoing challenge and there is always room for improvement. For example, when extending to real-life, large-scale multiagent systems with thousands of agents, how can we efficiently estimate the game-theoretic contribution of each agent or a group of agents?

- An opportunity related to the above challenge is to explore the use of additional application-specific information among the agents. For example, in an online social network with thousands of nodes, or an image with millions of pixel features, can we further exploit the structural relations among the agents (i.e., the nodes or the features)? Can the agents communicate with each other through a communication graph? We refer interested readers to some of the prior literature that studied efficient computation of semivalues over graphs [88, 136]. Furthermore, the graph neural network (GNN) is an emerging machine learning model which enables learning from graph-structured data. [153, 81, 65]. This provides a nice way of representing multiagent interactions: each node in the graph can represent an agent whose attributes are encoded as features, while edges in the graph can represent information between agents such as adjacency and types of relations. With the learning capabilities of graph neural networks, some new opportunities can be explored for computing payoff allocation for the agents, for example, learning to infer the value of a coalition based on the value of other related coalitions.

Many interesting future opportunities can be explored following our study on payoff allocation for submodular games and data valuation. A related application is post-hoc machine learning feature importance analysis. Much success of machine learning recently relies on deep neural networks (DNN) [71], an artificial neural network with multiple layers. On the one hand, the deep structure provides the capability for modeling many complex relationships through composing a large number of non-linear functions. On the other hand, the complex composition often leads to

the trained deep neural network lacking interpretability for humans. Interpretability is important for the model to be trustworthy [14]. For example, a seemingly accurate blackbox ML model may have exploited the spurious patterns in the datasets. Another example is when models can be easily attacked by adversarial parties even with small perturbations [137]. Moreover, interpretations of a learned ML model also serve as a good approach to knowledge discovery. Imagine that we now train a machine learning model for a quick COVID-19 diagnosis [156]. For each tested individual in the training dataset, the input can include a large variety of features such as age, contact with infected individuals, medical histories such as vaccination, exercise habits, and a selection of clinical symptoms such as fever and cough. An interpretability study of the model may rank the features by their importance towards the outcome of the record and help medical researchers identify the key features. In particular, feature selection problems are often considered and formulated as a submodular (or approximately submodular) problem, where adding more information is useful but yields diminishing returns due to the overlap of information. Moreover, replicated features have completely overlapping information, and hence can be suitably modelled with our replication-redundancy assumption. To this end, our current framework can be extended to study ML feature importance. Note that in the context of ML features, replication may not only come from a malicious manipulation, but can also be a result of duplicated records or overlapping information. An interesting topic would be to compare the importance measures of correlated or replicated features according to the different semivalues, and how different semivalues can help identify the key subset of features that best represent the ground set of features.

Apart from feature evaluation, we include some interesting future work in the following:

- Along the line of work of data valuation in multi-party machine learning, one can develop more criteria that evaluate the fairness of data valuation methods, examples include the robustness against replication, diversity, uniqueness, and coverage.

- Contrary to the post-hoc analysis, another future direction is to explore how the game-theoretic solution concepts can be integrated into the training loop to help guide the updates of machine learning models. For example, in a federated learning system, model updates often rely on the iterated learning process, where local nodes interact with a central server to perform learning in a decentralized manner. In each round of learning, the global model

is transmitted to the local nodes, followed by local model training, and central aggregation of the local updates towards the global model. An interesting direction is to evaluate the contribution of each local node using game-theoretic solution concepts, and adjust the subsequent training iterations according to their contributions.

- Apart from exploring alternative machine learning applications, an important future work is to expand our theoretical analysis of payoff allocation in submodular games. The following are some potential directions: 1. analysing metrics beyond the total payoff received by the replicating player, such as the individual payoff of each replica agent, the payoff of the honest players, and the cost of replication; 2. as suggested by our examples of data perturbation, and subset replication, one can expand the study of replications towards more complex manipulations; 3. relating to the seminal literature on splitting and merging, an interesting future direction is to study multi-lateral collusions in submodular games where multiple agents collude to obtain a higher collective payoff.

Many interesting and important future directions can be studied following our multiagent reinforcement credit assignment for robotic control.

- An immediate opportunity is to extend our model-based credit-assignment framework to alternative settings with simulated or real robots. Almost all real-life robotic control problems fall into the domain of continuous control and are interesting for future study, to list a few: formation control for a system of unmanned aerial vehicles (UAV); multiagent grasping and manipulation; multi-robot exploration of unknown environments, etc. Each of these sub-domains of robotic control may exhibit specific challenges which call for different coordination behaviours and credit assignment methods. Alternatively, the framework can also be applied to discrete multiagent domains such as Starcraft [42].

- Another possibility is to extend the study to alternative game-theoretic and reinforcement learning methods, for example: game-theoretic solution concepts beyond the semivalues; other continuous reinforcement learning algorithms such as deep deterministic policy gradient (DDPG) [77]; and alternative model-based reinforcement learning paradigms such as visual model-based algorithms which observe raw pixel inputs rather than state features [34].

- Algorithmically, a challenge in credit assignment for multiagent reinforcement learning is to optimise the system's global objective through decentralised agents' policies. In this regard, a future direction is to study how the game-theoretic payoff allocations to each agent affect their global objective; furthermore, it is interesting to theoretically study general credit assignment methods where the local rewards can guide the agents to learn coordinated policies which optimise the global objective.

- Finally, an interesting future direction is to study how various game-theoretic credit assignments as reward signals give rise to different roles and specialisations in the multiagent reinforcement learning system.

The above directions pose new challenges and fruitful opportunities to improve multiagent machine learning systems, and contribute towards the advancement of multiagent cooperation and coordination. In the future, we hope to see increasing research and real-world applications which integrate cooperative game theory and multiagent machine learning, providing multiagent systems with both powerful learning capabilities and principled theoretical foundations.

Appendices

A

Omitted Proofs for Chapter 3

Contents

A.1 Proofs for Section 3.3.2

Before deriving the Shapley and Banzhaf value for the facility location game, we first need to show the following mathematical identity which will be used for the derivation.

Lemma A.1.0.1.

$$\sum_{k=0}^{m} \frac{\binom{m}{k}}{\binom{n}{k}} = \frac{n+1}{n+1-m} \tag{A.1}$$

Proof Sketch. The identity can be shown in two steps: First, we show the identity $\frac{\binom{m}{k}}{\binom{n}{k}} = \frac{\binom{n-k}{m-k}}{\binom{n}{m}}$ by expansion of the terms. Then, we can take the denominator $\binom{n}{m}$ out of the summation over k, and as a common mathematical identity, the sum reduces to $\sum_{k=0}^{m} \binom{n-k}{m-k} = \binom{n+1}{m}$. Finally, by expanding the terms we arrive at $\frac{\binom{n+1}{m}}{\binom{n}{m}} = \frac{n+1}{n+1-m}$. □

Theorem 3.3.2.1. *The Shapley value of a facility location $i \in \mathcal{L}$ can be computed as*

$$\varphi_i^{\text{Shapley}} = \sum_{d \in D} \Big[u_{id} \frac{1}{|\mathcal{L}| - |\mathcal{L}_{id}|} - \sum_{t=1}^{|\mathcal{L}_{id}|} \frac{1}{\lambda(t) + \lambda(t)^2} u_{e_{it}^d} \Big],$$

where $\lambda(t) \coloneqq (|\mathcal{L}| - |\mathcal{L}_{id}| + t - 1)$, $\mathcal{L}_{id} \coloneqq \{j \in \mathcal{L} \setminus \{i\} \mid u_{jd} \leq u_{id}\}$ and $u_{e_{it}^d}$ is the utility value of the t-th largest element after i along the dimension (customer) d.

Proof. Let $v(C) \coloneqq Fac(C)$ and $n \coloneqq |\mathcal{L}|$ as the number of players. Denote w_C as the weights of the Shapley value, i.e., $\varphi_i^{\text{Shapley}} = \sum_{C \subseteq \mathcal{L} \setminus \{i\}} w_C \mathcal{MC}_i(C)$, where $w_C \coloneqq \frac{1}{n} \binom{n-1}{|C|}^{-1}$. Observe that

$$\varphi_i = \sum_{C \subseteq \mathcal{L} \setminus \{i\}} w_C \mathcal{MC}_i(C) \overset{(*)}{=} \sum_{d \in D} \Big[\underbrace{\sum_{C \subseteq \mathcal{L}_{id}} w_C u_{id}}_{\text{(\#1)}} - \underbrace{\sum_{C \subseteq \mathcal{L}_{id}} w_C \max_{j \in C} u_{jd}}_{\text{(\#2)}} \Big],$$

where $(*)$ is because the marginal contribution of i for dimension d is zero *unless* i is the largest element for that dimension and $\mathcal{L}_{id} = \{j \in \mathcal{L} \mid u_{jd} \leq u_{id}\}$ is the coalition of all elements which have smaller values in the d-th dimension than element i.

Along each dimension d, (#1) is a weighted sum over coalitions C where i is the largest element; and (#2) sums up for each $j \in \mathcal{L}_{id}$ over all coalitions $C \subseteq \mathcal{L}_{id}$ where j is the largest

element. We next compute (#1) and (#2) separately for each dimension $d \in D$:

$$
\begin{aligned}
(\#1) &= \sum_{C \subseteq \mathcal{L}_{id}} w_C u_{id} = u_{id} \sum_{C \subseteq \mathcal{L}_{id}} w_C \\
&= u_{id} \sum_{c=0}^{|\mathcal{L}_{id}|} \sum_{C \subseteq \mathcal{L}_{id}, |C|=c} w_C && \text{, where } w_C \coloneqq \frac{1}{n}\binom{n-1}{c}^{-1} \\
&= u_{id} \frac{1}{n} \sum_{c=0}^{|\mathcal{L}_{id}|} \binom{n-1}{c}^{-1} \binom{|\mathcal{L}_{id}|}{c} \\
&\overset{(1)}{=} u_{id} \frac{1}{n} \frac{n}{n - |\mathcal{L}_{id}|} && \text{, (1) by Lemma A.1.0.1 } \sum_{k=0}^{m} \frac{\binom{m}{k}}{\binom{n}{k}} = \frac{n+1}{n+1-m} \\
&= u_{id} \frac{1}{n - |\mathcal{L}_{id}|},
\end{aligned}
$$

Next we compute (#2). For simplicity, let $+, -$ denote set operations $C \cup \{e\}, C \setminus \{e\}$, and denote e_{it}^d is t-th largest element (after element i) in the d-th dimension.

$$
\begin{aligned}
(\#2) &= \sum_{C \subseteq \mathcal{L}_{id}} w_C \max_{j \in C} u_{jd} \\
&= \sum_{C \subseteq \mathcal{L}_{id} - (e_{i1}^d)} w_{C + e_{i1}^d} u_{e_{i1}^d d} + \sum_{C \subseteq \mathcal{L}_{id} - (e_{i1}^d + e_{i2}^d)} w_{C + e_{i2}^d} u_{e_{i2}^d d} + \cdots \\
&= u_{e_{i1}^d d} \sum_{C \subseteq \mathcal{L}_{id} - (e_{i1}^d)} w_{C + e_{i1}^d} + u_{e_{i2}^d d} \sum_{C \subseteq \mathcal{L}_{id} - (e_{i1}^d + e_{i2}^d)} w_{C + e_{i2}^d} + \cdots \\
&= u_{e_{i1}^d d} \underbrace{\sum_{C \subseteq \mathcal{L}_{id} - (e_{i1}^d)} w_{C + e_{i1}^d}}_{=:\beta_1} + u_{e_{i2}^d d} \underbrace{\sum_{C \subseteq \mathcal{L}_{id} - (e_{i1}^d + e_{i2}^d)} w_{C + e_{i2}^d}}_{=:\beta_2} + \cdots
\end{aligned}
$$

$$
\begin{aligned}
\text{In particular, } \beta_t &= \sum_{C \subseteq \mathcal{L}_{id} - (e^d_{i1} + \ldots + e^d_{it})} w_{C+e^d_{it}} \\
&= \sum_{c=0}^{|\mathcal{L}_{id}|-t} \sum_{C \subseteq \mathcal{L}_{id} - (e^d_{i1} + \ldots + e^d_{it}), |C|=c} w_{C+e^d_{it}} \\
&= \frac{1}{n} \sum_{c=0}^{|\mathcal{L}_{id}|-t} \binom{n-1}{c+1}^{-1} \binom{|\mathcal{L}_{id}| - t}{c} \\
&\overset{(1)}{=} \frac{1}{n} \sum_{c=0}^{|\mathcal{L}_{id}|-t} \binom{n-1}{c+1}^{-1} \left[\binom{|\mathcal{L}_{id}| - t + 1}{c+1} - \binom{|\mathcal{L}_{id}| - t}{c+1} \right] \\
&\overset{(2)}{=} \frac{1}{n} \sum_{x=1}^{|\mathcal{L}_{id}|-t+1} \binom{n-1}{x}^{-1} \left[\binom{|\mathcal{L}_{id}| - t + 1}{x} - \binom{|\mathcal{L}_{id}| - t}{x} \right] \\
&\overset{(3)}{=} \frac{1}{n} \left[\frac{n}{n - |\mathcal{L}_{id}| + t - 1} - 1 - \frac{n}{n - |\mathcal{L}_{id}| + t} + 1 \right] \\
&= \frac{1}{\lambda(t) + \lambda(t)^2},
\end{aligned}
$$

where (1) is by Pascal's identity, (2) by substituting $x = c+1$, (3) by Lemma A.1.0.1 and observing that $\binom{n}{k}$ is zero for $k > n$, and where $\lambda(t) = n - |\mathcal{L}_{id}| + t - 1$.

$$
\text{Hence, } \varphi_i = \sum_{d \in D} \left[u_{id} \frac{1}{n - |\mathcal{L}_{id}|} - \sum_{t=1}^{|\mathcal{L}_{id}|} \frac{1}{\lambda(t) + \lambda(t)^2} u_{e^d_{it}} \right]. \qquad \square
$$

Theorem 3.3.2.2. *The Banzhaf value of a facility location $i \in \mathcal{L}$ can be computed as*

$$
\varphi_i^{\text{Banzhaf}} = \frac{1}{2^{|\mathcal{L}|-1}} \sum_{d \in D} \left[2^{|\mathcal{L}_{id}|} u_{id} - \sum_{t=1}^{|\mathcal{L}_{id}|} 2^{|\mathcal{L}_{id}|-t} u_{e^d_{it}} \right],
$$

where $\mathcal{L}_{id} = \{j \in \mathcal{L} \setminus \{i\} \mid u_{jd} \leq u_{id}\}$ and $u_{e^d_{it}}$ is the utility value of the t-th largest element after i for dimension d.

Proof. Similar to the proof for the Shapley value, we expand the Banzhaf value as follows:

$$
\varphi(i) = \sum_{C \subseteq \mathcal{L} - i} w_C \mathcal{MC}_i(C) = \sum_{d \in D} \Big[\underbrace{\sum_{C \subseteq \mathcal{L}_{id}} w_C u_{id}}_{(\#1)} - \underbrace{\sum_{C \subseteq \mathcal{L}_{id}} w_C \max_{j \in C} u_{jd}}_{(\#2)} \Big].
$$

By definition of the Banzhaf value $w_C \coloneqq \frac{1}{2^{n-1}}$, next we compute #1 and #2.

$$\begin{aligned}
(\#1) &= \sum_{d\in D}\sum_{C\subseteq\mathcal{L}_{id}} w_C u_{id} \\
&= \sum_{d\in D} u_{id} \sum_{C\subseteq\mathcal{L}_{id}} w_C \\
&= \sum_{d\in D} u_{id} \sum_{c=0}^{|\mathcal{L}_{id}|} \sum_{C\subseteq\mathcal{L}_{id},|C|=c} w_C \\
&= \sum_{d\in D} u_{id} \frac{1}{2^{n-1}} \sum_{c=0}^{|\mathcal{L}_{id}|} \binom{|\mathcal{L}_{id}|}{c} \\
&= \frac{1}{2^{n-1}} \sum_{d\in D} 2^{|\mathcal{L}_{id}|} u_{id}
\end{aligned}$$

We then expand (#2) in a similar approach to the Shapley value (for notations c.f. above theorem),

$$(\#2) = \sum_{C\subseteq\mathcal{L}_{id}} w_C \max_{j\in C} u_{jd} = u_{e^d_{i1}d} \underbrace{\sum_{C\subseteq\mathcal{L}_{id}-(e^d_{i1})} w_{C+e^d_{i1}}}_{=:\beta_1} + u_{e^d_{i2}d} \underbrace{\sum_{C\subseteq\mathcal{L}_{id}-(e^d_{i1}+e^d_{i2})} w_{C+e^d_{i2}}}_{=:\beta_2} + \cdots$$

$$\begin{aligned}
\text{where } \beta_t &= \sum_{C\subseteq\mathcal{L}_{id}-(e^d_{i1}+\ldots+e^d_{it})} w_{C+e^d_{it}} \\
&= \sum_{c=0}^{|\mathcal{L}_{id}|-t} \sum_{C\subseteq\mathcal{L}_{id}-(e^d_{i1}+\ldots+e^d_{it}),|C|=c} w_{C+e^d_{it}} \\
&= \frac{1}{2^{n-1}} \sum_{c=0}^{|\mathcal{L}_{id}|-t} \binom{|\mathcal{L}_{id}|-t}{c} \\
&= \frac{1}{2^{n-1}} 2^{|\mathcal{L}_{id}|-t}
\end{aligned}$$

$$\text{Hence, } \varphi_i = \frac{1}{2^{n-1}} \sum_{d\in D} \Big[2^{|\mathcal{L}_{id}|} u_{id} - \sum_{t=1}^{|\mathcal{L}_{id}|} 2^{|\mathcal{L}_{id}|-t} u_{e^d_{it}}\Big]. \qquad \square$$

A.2 Proofs for Section 3.3.4

Lemma 3.3.4.1. *Let G= (N, v) be a submodular game with replication-redundant characteristic function v, a player $i \in N$ replicates and obtains the total payoff as two identities $C^R = \{i_1, i_2\}$ in the new game $G^R = (N^R, v^R)$, where the players $N^R = N\setminus\{i\}\cup C^R$, and the induced characteristic function v^R satisfies $\forall C \subseteq N \setminus \{i\}\ \forall i_k \in C^R\colon v^R(i_k \cup C) = v(i \cup C)$ and $v^R(C) = v(C)$. By replicating, the changes in total payoff of player i is:*

$$\delta\varphi_i^{\text{Shapley}} = \sum_{C\subseteq N\setminus\{i\}} \frac{|C|!(|N|-|C|-1)!}{(|N|+1)!}(|N|-2|C|-1)\mathcal{MC}_i(C)$$

Proof. In the induced game $G^R = (N^R, v^R)$ where player i replicates into two players $\{i_1, i_2\}$, the total number of players increases by one, i.e., $|N^R| = |N| + 1$ and $v^R(C \cup \{i_1, i_2\}) = v^R(C \cup \{i\})$ as i_1 and i_2 are replicas of i and as a result of replication redundancy. We next write out the sum of the Shapley values $\varphi^R_{i_2}$ and $\varphi^R_{i_2}$ of i_1, i_2 in G^R.

$$\begin{aligned}
\varphi^R_{i_1}(N^R, v^R) &\coloneqq \sum_{C \subseteq N^R \setminus \{i\}} \frac{|C|!(|N^R| - |C| - 1)!}{|N^R|!} \mathcal{MC}_{i_1}(C) \\
&= \sum_{C \subseteq N^R \setminus \{i\}} \frac{|C|!(|N| + 1 - |C| - 1)!}{(|N| + 1)!} \mathcal{MC}_{i_1}(C) \\
&\overset{(1)}{=} \sum_{C \subseteq N^R \setminus \{i_1, i_2\}} \frac{|C|!(|N| - |C|)!}{(|N| + 1)!} \mathcal{MC}_{i_1}(C) + \sum_{C \subseteq N^R \setminus \{i_1, i_2\}} \frac{(|C| + 1)!(|N| - |C| - 1)!}{(|N| + 1)!} \underbrace{\mathcal{MC}_{i_1}(C \cup \{i_2\})}_{= 0, \text{ replication redundancy}} \\
&= \underbrace{\sum_{C \subseteq N^R \setminus \{i_1, i_2\}}}_{= N \setminus \{i\}} \frac{|C|!(|N| - |C|)!}{(|N| + 1)!} \underbrace{\mathcal{MC}_{i_1}(C)}_{= \mathcal{MC}_i(C)} = \sum_{C \subseteq N \setminus \{i\}} \frac{|C|!(|N| - |C|)!}{(|N| + 1)!} \mathcal{MC}_i(C),
\end{aligned}$$

where (1) is by grouping the coalitions of players (excluding i_1) into two groups, one group containing all coalitions without i_2 and the other with i_2 in.

By symmetry, $\varphi^R_{i_2}(N^R, v^R) = \varphi^R_{i_1}(N^R, v^R)$, and the total payoff of i in the induced game is:

$$\varphi^R_i = 2\varphi^R_{i_1}(N^R, v^R) = \sum_{C \subseteq N \setminus \{i\}} \frac{2|C|!(|N| - |C|)!}{(|N| + 1)!} \mathcal{MC}_i(C)$$

On the other hand, the total payoff of player i in the original game is:

$$\varphi_i = \sum_{C \subseteq N \setminus \{i\}} \frac{|C|!(|N| - |C| - 1)!}{|N|!} \mathcal{MC}_i(C)$$

Therefore, the change in total payoff of player i is :

$$\begin{aligned}
\delta\varphi_i^{\text{Shapley}} &= \varphi^R_i - \varphi_i \\
&= \sum_{C \subseteq N \setminus \{i\}} \frac{|C|!(|N| - |C| - 1)!}{|N| + 1!} (2(|N| - |C|) - (|N| + 1)) \mathcal{MC}_i(C) \\
&= \sum_{C \subseteq N \setminus \{i\}} \frac{|C|!(|N| - |C| - 1)!}{|N| + 1!} (|N| - 2|C| - 1) \mathcal{MC}_i(C)
\end{aligned}$$

□

Lemma 3.3.4.3. *Let G= (N, v) be a submodular game with replication-redundant characteristic function v, a player $i \in N$ replicates and obtains the total payoff as two identities $C^R = \{i_1, i_2\}$ in the new game $G^R = (N^R, v^R)$, where the players are $N^R = N \setminus \{i\} \cup C^R$, and v^R satisfies*

$\forall C \subseteq N \setminus \{i\}\ \forall i_k \in C^R\colon v^R(i_k \cup C) = v(i \cup C)$ *and* $v^R(C) = v(C)$. *Under payoff allocation using the Banzhaf value, the changes in total payoff of player* i *by replicating is zero, i.e.,* $\delta\varphi_i^{\text{Banzhaf}} = 0$

Proof. In the induced game $G^R = (N^R, v^R)$ where player i replicates into two players $\{i_1, i_2\}$, the total number of players increases by one, i.e., $|N^R| = |N| + 1$ and $v^R(C \cup \{i_1, i_2\}) = v^R(C \cup \{i\})$ as i_1 and i_2 are replicas of i and as a result of replication redundancy. We next write out the sum of the Shapley values $\varphi_{i_2}^R$ and $\varphi_{i_2}^R$ of i_1, i_2 in G^R.

$$
\begin{aligned}
\varphi_{i_1}^R(N^R, v^R) &\coloneqq \sum_{C \subseteq N^R \setminus \{i\}} \frac{1}{2^{|N^R|}} \mathcal{MC}_{i_1}(C) \\
&= \sum_{C \subseteq N^R \setminus \{i\}} \frac{1}{2^{|N|+1}} \mathcal{MC}_{i_1}(C) \\
&\overset{(1)}{=} \sum_{C \subseteq N^R \setminus \{i_1, i_2\}} \frac{1}{2^{|N|+1}} \mathcal{MC}_{i_1}(C) + \sum_{C \subseteq N^R \setminus \{i_1, i_2\}} \frac{1}{2^{|N|+1}} \underbrace{\mathcal{MC}_{i_1}(C \cup \{i_2\})}_{=0,\ \text{replication redundancy}} \\
&= \underbrace{\sum_{C \subseteq N^R \setminus \{i_1, i_2\}}}_{=N \setminus \{i\}} \frac{1}{2^{|N|+1}} \underbrace{\mathcal{MC}_{i_1}(C)}_{=\mathcal{MC}_i(C)} = \sum_{C \subseteq N \setminus \{i\}} \frac{1}{2^{|N|+1}} \mathcal{MC}_i(C),
\end{aligned}
$$

where (1) is by grouping the coalitions of players (excluding i_1) into two groups, one group containing all coalitions without i_2 and the other with i_2 in.

By symmetry, $\varphi_{i_2}^R(N^R, v^R) = \varphi_{i_1}^R(N^R, v^R)$, and the total payoff of i in the induced game is:

$$
\varphi_i^R = 2\varphi_{i_1}^R(N^R, v^R) = \sum_{C \subseteq N \setminus \{i\}} \frac{1}{2^{|N|}} \mathcal{MC}_i(C) = \varphi_i(N, v)
$$

Therefore, the change in total payoff of player i is zero, i.e.,:

$$
\delta\varphi_i^{\text{Banzhaf}} = \varphi_i^R - \varphi_i = 0
$$

□

A.3 Proof for Section 3.3.5

Lemma 3.3.5.2. *Let* $G = (N, v)$ *be a submodular game with replication-redundant characteristic function* v. *By replicating* k *times and acting as* $k + 1$ *players* $C^R = \{i_0, \ldots, i_k\}$ *in the induced*

game $G^R = (N^R, v^R)$, the malicious player i receives a total payoff of

$$\varphi_i^{\text{tot}}(k) = \sum_{c=0}^{|N|-1} \alpha_c^k z_i(c), \text{ where} \tag{3.5}$$

$$z_i(c) = \binom{|N|-1}{c}^{-1} \sum_{C \subseteq N\setminus\{i\}, |C|=c} \mathcal{MC}_i(C) \quad \text{(avg. marginal contributions in } G\text{)},$$

$$\alpha_c^k = (k+1)\binom{|N|-1}{c} w_{c,N^R} \quad \text{(importance weights after replication)}.$$

Proof. In the induced game G^R, let $w_{|C|,N+k}$ be the weights of the solution concept by definition, i.e., $\varphi_i(N^R, v^R) \coloneqq \sum_{C \subseteq N^R\setminus\{i\}} w_{|C|,N+k}\mathcal{MC}_i(C)$. Then the total payoff of the malicious player after k replications is:

$$\begin{aligned}
\varphi_i^{\text{tot}}(k) &\overset{(1)}{=} (k+1)\varphi_{i_k}(N^R, v^R) \\
&= (k+1) \sum_{C \subseteq N^R\setminus\{i_k\}} w_{|C|,N+k}\mathcal{MC}_{i_k}(C) \\
&= (k+1) \sum_{C \subseteq N^R\setminus C_R} w_{|C|,N+k}\mathcal{MC}_{i_k}(C) + (k+1) \sum_{C \subseteq N^R\setminus\{i_k\}, C\cap C_R \neq \phi} w_{|C|,N+k} \underbrace{\mathcal{MC}_{i_k}(C)}_{\overset{(2)}{=}0} \\
&= (k+1) \sum_{S \subseteq N\setminus\{i\}} w_{|C|,N+k}\mathcal{MC}_i(C) \\
&= \sum_{c=0}^{N-1} \underbrace{(k+1)\binom{N-1}{c} w_{|C|,N+k}}_{\alpha_c^k} z_i(c)
\end{aligned}$$

where (1) is due to symmetry and (2) is due to replication-redundancy. □

A.4 Proofs for Section 3.3.6.3

Lemma 3.3.6.1. *For the Shapley value, the importance weights α_c^k of the total payoff of a replicating player satisfy the following properties: $\forall k \geq 0, \forall 0 \leq p \leq |N| - 1$,*

$$\sum_{c=0}^{|N|-1} \alpha_c^k = 1 \tag{3.9a}$$

$$\sum_{c=0}^{p} \alpha_c^k \leq \sum_{c=0}^{p} \alpha_c^{k+1} \tag{3.9b}$$

$$\sum_{c=0}^{p} \alpha_c^{k+1} - \alpha_c^k \geq \sum_{c=0}^{p} \alpha_c^{k+2} - \alpha_c^{k+1} \tag{3.9c}$$

Proof. **Proof for Equation (3.9a):** Equation (3.9a) shows that α_c^k always sums to 1 under

changing k for the Shapley value.

$$\begin{aligned}
\sum_{c=0}^{N-1} \alpha_c^k &= \sum_{c=0}^{N-1} \frac{k+1}{N+k} \binom{N-1}{c} \binom{N+k-1}{c}^{-1}, \text{ due to Corollary 3.3.5.1} \\
&= (k+1) \sum_{c=0}^{N-1} \frac{(N-1)!(N+k-1-c)!}{(N-1-c)!(N+k)!} \\
&= \frac{(k+1)!(N-1)!}{(N+k)!} \sum_{c=0}^{N-1} \frac{(N+k-1-c)!}{(N-1-c)!k!} \\
&= \frac{1}{\binom{N+k}{k+1}} \sum_{c=0}^{N-1} \binom{N+k-1-c}{k} \\
&\overset{(1)}{=} \frac{1}{\binom{N+k}{k+1}} \sum_{i=k}^{N-k-1} \binom{i}{k} \\
&\overset{(2)}{=} \frac{1}{\binom{N+k}{k+1}} \binom{N+k}{k+1} \\
&= 1,
\end{aligned}$$

where (1) is by substituting $i = N + k - 1 - c$ and (2) by the Hockey-Stick identity.

Proof for Equation (3.9b): This shows that α_c^k *shift* to the smaller coalitions under growing k.

$$\begin{aligned}
\sum_{c=0}^{p} \alpha_c^k &= \sum_{c=0}^{p} \frac{(k+1)(N-1)!(N+k-1-c)!}{(N-1-c)!(N+k)!} \\
&= \frac{(k+1)!(N-1)!}{(N+k)!} \sum_{c=0}^{p} \frac{(N+k-1-c)!}{(N-1-c)!k!} \\
&= \frac{1}{\binom{N+k}{k+1}} \sum_{c=0}^{p} \binom{N+k-1-c}{k} \\
&= \frac{1}{\binom{N+k}{k+1}} \Big(\sum_{c=0}^{N-1} - \sum_{c=p+1}^{N-1} \binom{N+k-1-c}{k} \Big) \\
&\overset{(1)}{=} 1 - \frac{1}{\binom{N+k}{k+1}} \sum_{c=p+1}^{N-1} \binom{N+k-1-c}{k} \\
&\overset{(2)}{=} 1 - \frac{1}{\binom{N+k}{k+1}} \binom{N+k-p-1}{k+1} \\
&= 1 - \frac{(N-1)!}{(N-p-2)!} \frac{1}{(N+k)...(N+k-p)},
\end{aligned}$$

where (1) is by Equation (3.9a) and (2) is by the Hockey-Stick identity. Similarly,

$$\sum_{c=0}^{p} \alpha_c^{k+1} = 1 - \frac{(N-1)!}{(N-p-2)!} \frac{1}{(N+k+1)...(N+k+1-p)}$$

Therefore,

$$\sum_{c=0}^{p} \alpha_c^{k+1} - \sum_{c=0}^{p} \alpha_c^{k} = \frac{(N-1)!}{(N-p-2)!} \frac{(N+k+1)-(N+k-p)}{(N+k+1)...(N+k-p)}$$
$$= \frac{(N-1)!}{(N-p-2)!} \frac{p+1}{(N+k+1)...(N+k-p)} \geq 0$$

And this concludes our proof for Equations (3.9a) and (3.9b)

(Additional) Proof for Equation (3.9c): With this additional condition, the *unit gain* of the total payoff for adding a replica decreases monotonically for each replication. This means that the player obtains the most unit gain for the first replication. To show this property, we denote for any k, denote $\delta^k := \sum_{c=0}^{p} \alpha_c^{k+1} - \sum_{c=0}^{p} \alpha_c^{k}$. From the proof of Equation (3.9b): $\delta^k = \frac{(N-1)!}{(N-p-2)!} \frac{p+1}{(N+k+1)\dots(N+k-p)}$, therefore,

$$RHS - LHS \text{ of (3.9c)} = \delta^{k+1} - \delta^k$$
$$= \frac{(N-1)!(p+1)}{(N-p-2)!} \left(\frac{1}{(N+k+2)...(N+k+1-p)} - \frac{1}{(N+k+1)...(N+k-p)}\right)$$
$$= \frac{(N-1)!(p+1)}{(N-p-2)!} \left(\frac{(N+k-p)-(N+k+2)}{(N+k+2)...(N+k-p)}\right)$$
$$= \frac{(N-1)!(p+1)}{(N-p-2)!} \frac{-(p+2)}{(N+k+2)...(N+k-p)} \leq 0 \qquad \square$$

A.5 Proofs for Section 3.3.6.4

Corollary 3.3.6.3. *The Robust Shapley value in Definition 3.3.6.1 is replication-robust. Moreover, in a submodular game $G = (N, v)$, the loss for a replicating player i by replicating k times is at least: $\varphi_i^{\text{tot}}(0) - \varphi_i^{\text{tot}}(k) \geq \frac{1}{|N|} \sum_{c=0}^{|N|-1} (1 - \frac{k+1}{2^k}) \gamma_{|N|}^{c} z_i(c)$.*

Proof. **Replication-robustness** We prove that similar to the Banzhaf value, the Robust Shapley value satisfies Equation (3.10) in Observation 1: $\forall k \geq 0, \frac{\alpha_c^k}{\alpha_c^{k+1}} \geq 1$. Hence it satisfies Theorem 3.3.6.1, and therefore sufficient for replication-robustness. There are 3 possible cases:

Case 1: $c < \lfloor \frac{N+k-1}{2} \rfloor \leq \lfloor \frac{N+k}{2} \rfloor$

In this case, both $\tilde{\alpha}_c^k$ and $\tilde{\alpha}_c^{k+1}$ will be down-weighed from the Shapley coefficients where

$\gamma^c_{N+k} = \frac{\lceil \frac{N+k-1}{2} \rceil ! \lfloor \frac{N+k-1}{2} \rfloor !}{c!(N+k-c-1)!}$:

$$\tilde{\alpha}^k_c = \gamma^c_{N+k}\alpha^k_c = (k+1)\binom{N-1}{c}\frac{\lfloor \frac{N+k-1}{2} \rfloor ! \lceil \frac{N+k-1}{2} \rceil !}{(N+k)!}$$
$$\tilde{\alpha}^{k+1}_c = \gamma^c_{N+k+1}\alpha^{k+1}_c = (k+2)\binom{N-1}{c}\frac{\lfloor \frac{N+k}{2} \rfloor ! \lceil \frac{N+k}{2} \rceil !}{(N+k+1)!}$$
$$\text{Hence } \frac{\tilde{\alpha}^k_c}{\tilde{\alpha}^{k+1}_c} = \frac{k+1}{k+2}\frac{N+k+1}{\lceil \frac{N+k}{2} \rceil} \geq \frac{1}{2} * 2 = 1.$$

Case 2: $c \geq \lfloor \frac{N+k}{2} \rfloor \geq \lfloor \frac{N+k-1}{2} \rfloor$

Both $\tilde{\alpha}^k_c, \tilde{\alpha}^{k+1}_c$ take the original form of Shapley coefficients after replication, i.e, $\gamma^c_N = 1$:

$$\tilde{\alpha}^k_c = \alpha^k_c = (k+1)\binom{N-1}{c}\frac{c!(N+k-1-c)!}{(N+k)!}$$
$$\tilde{\alpha}^{k+1}_c = \alpha^{k+1}_c = (k+2)\binom{N-1}{c}\frac{c!(N+k-c)!}{(N+k+1)!}$$
$$\frac{\tilde{\alpha}^k_c}{\tilde{\alpha}^{k+1}_c} = \frac{k+1}{k+2}\frac{N+k+1}{N+k-c} \overset{(1)}{\geq} 2\frac{k+1}{k+2} \geq 1, \quad \text{where (1) is due to } c \geq \lfloor \tfrac{N+k}{2} \rfloor.$$

Case 3: $\lfloor \frac{N+k-1}{2} \rfloor \leq c < \lfloor \frac{N+k}{2} \rfloor$

In this case, $\tilde{\alpha}^k_c$ will take the original form, while $\tilde{\alpha}^{k+1}_c$ will take the down-weighed form. Moreover, $N+k$ must be even, hence $c = \lfloor \frac{N+k-1}{2} \rfloor$.

$$\tilde{\alpha}^k_c = \alpha^k_c = (k+1)\binom{N-1}{c}\frac{c!(N+k-1-c)!}{(N+k)!} = (k+1)\binom{N-1}{c}\frac{\lfloor \frac{N+k-1}{2} \rfloor ! \lceil \frac{N+k-1}{2} \rceil !}{(N+k)!}$$
$$\tilde{\alpha}^{k+1}_c = \gamma^c_{N+k+1}\alpha^{k+1}_c = (k+2)\binom{N-1}{c}\frac{\lfloor \frac{N+k}{2} \rfloor ! \lceil \frac{N+k}{2} \rceil !}{(N+k+1)!}$$
$$\text{Hence } \frac{\tilde{\alpha}^k_c}{\tilde{\alpha}^{k+1}_c} = \frac{k+1}{k+2}\frac{N+k+1}{\lceil \frac{N+k}{2} \rceil} \geq 2\frac{k+1}{k+2} \geq 1$$

We have shown that $\forall k \geq 0$, $\frac{\alpha^k_c}{\alpha^{k+1}_c} \geq 1$, and hence the Robust Shapley value is replication-robust.

Payoff loss Note that from the above derivations, in all 3 cases, $\forall k \geq 0$, $\frac{k+2}{k+1}\frac{\tilde{\alpha}^k_c}{\tilde{\alpha}^{k+1}_c} \geq 2$:

$$
\begin{aligned}
\varphi_i^{\text{tot}}(0) &= \sum_{c=0}^{N-1} \tilde{\alpha}_c^0 z_i(c) \coloneqq \frac{1}{N} \sum_{c=0}^{N-1} \gamma_N^c z_i(c) \\
\varphi_i^{\text{tot}}(k) &= \sum_{c=0}^{N-1} \tilde{\alpha}_c^k z_i(c) \\
&= (k+1) \sum_{c=0}^{N-1} \frac{\tilde{\alpha}_c^k}{k+1} z_i(c), \text{ and as } \forall k \geq 0, \frac{\tilde{\alpha}_c^{k}/(k+1)}{\tilde{\alpha}_c^{k+1}/(k+2)} \geq 2, \\
&\leq (k+1) \sum_{c=0}^{N-1} \frac{1}{2} \frac{\tilde{\alpha}_c^{k-1}}{k} z_i(c) \leq \ldots \\
&\leq (k+1) \sum_{c=0}^{N-1} \frac{1}{2^k} \tilde{\alpha}_c^0 z_i(c) \\
&= (\frac{k+1}{2^k}) \frac{1}{N} \sum_{c=0}^{N-1} \gamma_N^c z_i(c) \\
\textit{Hence } \varphi_i^{\text{tot}}(0) - \varphi_i^{\text{tot}}(k) &\geq \frac{1}{N} \sum_{c=0}^{N-1} (1 - \tfrac{k+1}{2^k}) \gamma_N^c z_i(c).
\end{aligned}
$$

This concludes our proof for Corollary 3.3.6.3. □

A.6 Replication Robustness of Binomial Semivalues for Section 3.3.6.4

To illustrate this effect of replication robustness vs. the individual/complementary value (c.f., section 3.3.6.4), we will discuss the **binomial semivalues** [17], which are a Banzhaf-like family of semivalues that are defined by the binomial distribution parameter p, i.e., the probability of a successful on a single Bernoulli trial $p \in [0, 1]$,

$$
\varphi_i^{\text{binomial}}(N, v) = \sum_{C \subseteq N \setminus \{i\}, c=|C|} w_{c,N} \mathcal{MC}_i(C), \text{ where } \quad w_{c,N} = p^c (1-p)^{|N|-c-1}
$$

The following equation represents a p-binomial semivalue according the average marginal contributions,

$$
\begin{aligned}
\varphi_i^{\text{binomial}} &= \sum_{c=0}^{|N|-1} \alpha_c z_i(c), \text{ where} \\
\alpha_c &= p^c (1-p)^{|N|-c-1} \tbinom{|N|-1}{c} \\
z_i(c) &= \tbinom{|N|-1}{c}^{-1} \sum_{C \subseteq N \setminus \{i\}, |C|=c} \mathcal{MC}_i(C)
\end{aligned}
$$

Notice that the importance weights here form a binomial distribution:

Consider a random variable X, which follows the binomial distribution with parameters $n = |N| - 1$, and $p \in [0, 1]$, we write $X \sim B(|N| - 1, p)$. Intuitively, if we throw a fair coin

where, $p = \frac{1}{2}$, the importance weights $\alpha_c = \frac{1}{2^{|N|-1}}\binom{N-1}{c}$ is the distribution of the number of times that we get a head. In general, given the probability parameter $p \in [0, 1]$, the probability mass function is defined by

$$P(n, p, c) = \binom{n}{c} p^c (1-p)^{n-c} \tag{A.2}$$

Therefore, by varying the parameter p, we can obtain a family of binomial semivalues such as the Banzhaf value (where $p = \frac{1}{2}$), i.e.,

$$\alpha_c = \binom{N-1}{c} p^c (1-p)^{N-1-c} \tag{A.3}$$

Robustness of the Binomial Semivalues: Let's now take a look at the robustness properties of the binomial semivalues against replication. To do this, recall the total payoff of the malicious player after replication:

$$\varphi_i^{\text{tot}}(k) = \sum_{c=0}^{|N|-1} \alpha_c^k z_i(c), \text{ where} \tag{A.4}$$

$$\alpha_c^k = (k+1)\binom{|N|-1}{c} w_{c,N^R} \quad \text{(New Importance Weights)} \tag{A.5}$$

$$z_i(c) = \frac{\sum_{|C|=c, C\subseteq N\setminus\{i\}}}{\binom{|N|-1}{c}} \qquad \text{(Original Average Marginal Contributions)} \tag{A.6}$$

Given a binomial semivalue defined by the probability parameter p,

$$\alpha_c(N) = \binom{|N|-1}{c} p^c (1-p)^{|N|-1-c} \tag{A.7}$$

Now let's compute the new importance weights. First, we need to get w_{c,N^R} where $|N^R| = |N| + k$ is the new number of players after the malicious player replicates k times. As the importance weights in the new game follow the binomial distribution, we have

$$\alpha_c(N^R) = \binom{|N|+k-1}{c} p^c (1-p)^{|N|+k-1-c} \tag{A.8}$$

Hence the weights of the semivalue towards each size c coalition are given by

$$w_{c,N^R} = \frac{1}{\binom{|N|+k-1}{c}} \alpha_c(N^R) = p^c (1-p)^{|N|+k-1-c} \tag{A.9}$$

Therefore, the new importance weights α_c^k are:

$$\alpha_c^k = (k+1)\binom{|N|-1}{c} w_{c,N^R} \tag{A.10}$$

$$= (k+1)\binom{|N|-1}{c} p^c (1-p)^{|N|+k-1-c} \tag{A.11}$$

Now we can compare the new importance weights with the original importance weights:

$$\alpha_c^k = (k+1)\binom{|N|-1}{c} p^c (1-p)^{|N|+k-1-c} \tag{A.12}$$

$$\alpha_c^0 = \binom{|N|-1}{c} p^c (1-p)^{|N|-1-c} \tag{A.13}$$

where the ratio can be obtained as

$$\frac{\alpha_c^k}{\alpha_c^0} = (k+1)(1-p)^k \tag{A.14}$$

Since $1-p < 1$, α_c^k will converge towards 0 as k goes to infinity. Therefore, all binomial semivalues will eventually converge to 0.

Robustness of the Binomial Semivalues When $k = 1$: Though the family of binomial semivalues display the same convergence behaviour (eventually replication robust as k increases), they have different degree of robustness when k is small, for example, when $k = 1$, the Banzhaf value is the only *neutral* binomial semivalue, while $p > \frac{1}{2}$ leads to a family of replication robust semivalue and $p < \frac{1}{2}$ a family non robust semivalue.

Observe that when $k = 1$,

$$\frac{\alpha_c^k}{\alpha_c^0} = (k+1)(1-p)^k = 2(1-p) \tag{A.15}$$

- If $p = \frac{1}{2}$, i.e., the Banzhaf value, $\frac{\alpha_c^k}{\alpha_c^0} = 1$, and that is the underlying reason of the Banzhaf value being neutral to a single replication.
- If $p > \frac{1}{2}$, $\frac{\alpha_c^k}{\alpha_c^0} < 1$, and the total payoff of the replicating player decreases. Hence the binomial semivalue is robust.
- If $p < \frac{1}{2}$, $\frac{\alpha_c^k}{\alpha_c^0} > 1$, and the total payoff of the replicating player increases. Hence the binomial semivalue is not robust. Furthermore, with growing k, the total payoff of the malicious player will first increase, then decrease and converges to 0.

More intuitively with the coin toss example, $p > \frac{1}{2}$ implies that it is more likely to get a head, resulting in a skewed distribution where the importance weights favours the larger coalitions (i.e., complementary value), and in comparison it is more robust than $p < \frac{1}{2}$, where the semivalue favours the smaller coalitions (i.e., individual value). Moreover, when k is large, the changes in the distribution of the importance weights play a major role (e.g., whether the new importance weights are shifting towards smaller/larger coalitions).

A.7 Proofs for Section 3.4.2

Lemma 3.4.2.1. *Compared with replication, the additional gain in total payoff of the malicious player due to the perturbation when replicating k times is given by:*

$$\varphi_i^{\text{perturb}} - \varphi_i^{\text{replicate}} \leq (k+1)\epsilon.$$

Proof. Compared with replication, the additional gain in payoff due to the perturbation is

$$\begin{aligned}
\varphi_i^{\text{perturb}} - \varphi_i^{\text{replicate}} &= \sum_{p_k \in C^P} \sum_{C \subseteq N\setminus\{i\}, C^p \subseteq C^P \setminus \{p_k\}} w_{|C\cup C^p|, N+k} \mathcal{MC}_{p_k}(C \cup C^p) - \\
&\qquad \sum_{i_k \in C^R} \sum_{C \subseteq N\setminus\{i\}, C^r \subseteq C^R \setminus \{i_k\}} w_{|C\cup C^r|, N+k} \mathcal{MC}_{i_k}(C \cup C^r) \\
&= \sum_{k=0}^{|C^R|-1} \sum_{C \subseteq N\setminus\{i\}} w_{|C|, N+k} (\mathcal{MC}_{p_k}(C) - \mathcal{MC}_{i_k}(C)) + \\
&\qquad \sum_{p_k \in C^P} \sum_{C \subseteq N\setminus\{i\}, C^p \subseteq_{\neq \emptyset} C^P \setminus \{p_k\}} w_{|C\cup C^p|, N+k} \mathcal{MC}_{p_k}(C \cup C^p) \\
&= \sum_{p_k \in C^P} \sum_{C \subseteq N\setminus\{i\}, C^p \subseteq_{\neq \emptyset} C^P \setminus \{p_k\}} w_{|C\cup C^p|, N+k} \mathcal{MC}_{p_k}(C \cup C^p) \\
&\overset{(1)}{\leq} (k+1) \sum_{C \subseteq N\setminus\{i\}, C^p \subseteq_{\neq \emptyset} C^P \setminus \{p_k\}} w_{|C\cup C^p|, N+k} \epsilon \\
&\overset{(2)}{\leq} (k+1)\epsilon,
\end{aligned}$$

where (1) is due to the assumption on ϵ and submodularity, where $\forall_{p_j \in C^p}, C \subseteq N \setminus \{i\}, \mathcal{MC}_{p_k}(C \cup C^p) \leq \epsilon$. (2) is due to the definition of semivalues where the weights of coalitions sum to 1. □

Lemma 3.4.2.2. *Under payoff allocation using the Shapley value, the total payoff to a malicious player by replication is no less than subset replication and splitting.*

Proof. Due to the efficiency axiom of the Shapley value, the payoff of all (malicious and honest) players sum to the value of the grand coalition, where all data are pooled and is hence equal across all three attack scenarios, i.e., $\varphi_{\text{tot}} + \sum_{j \in N\setminus\{i\}} \varphi_j = v(N)$, where φ_{tot} is the total payoff of the malicious player, and j are the other (honest) players. Therefore, to show that replication yields the largest total payoff for the malicious player, we only need to show the honest players receive less payoff when i replicates (than splitting or replicating subsets). We show that this is true as a result of submodularity: For any honest player $j \in N \setminus \{i\}$,

$$\varphi_j = \sum_{C \subseteq N\setminus\{i,j\}, C^r \subseteq C^R} w_{|C\cup C^r|, N+k} \mathcal{MC}_j(C \cup C^r)$$

In this equation, for each identity $i_k \in S^r$, the data held due to splitting and subset replication are a subset of the data held due to replication:

$$D_{i_k^{\text{split}}} \subseteq D_{i_k^{\text{rep}}} \text{ and } D_{i_k^{\text{sub}}} \subseteq D_{i_k^{\text{rep}}}, \text{ hence } \begin{cases} \bigcup_{i^k \in C^r} D_{i_k^{\text{split}}} \subseteq \bigcup_{i^k \in C^r} D_{i_k^{\text{rep}}}; \\ \bigcup_{i^k \in C^r} D_{i_k^{\text{sub}}} \subseteq \bigcup_{i^k \in C^r} D_{i_k^{\text{rep}}}. \end{cases}$$

Therefore, due to submodularity, the marginal contribution of player j against replicating identities is less than splitting and subset replications: $\forall C^r \subseteq C^R$

$$\begin{aligned} \mathcal{MC}_j(C \cup \bigcup_{i^k \in C^r} D_{i_k^{\text{rep}}})) &\leq \mathcal{MC}_j(C \cup \bigcup_{i^k \in C^r} D_{i_k^{\text{split}}}) \\ \mathcal{MC}_j(C \cup \bigcup_{i^k \in C^r} D_{i_k^{\text{rep}}})) &\leq \mathcal{MC}_j(C \cup \bigcup_{i^k \in C^r} D_{i_k^{\text{sub}}}) \end{aligned}$$

Therefore, the payoff of an honest player φ_j against the replicating player is smallest. And hence using the efficiency axiom, $\varphi_{\text{tot}}^{\text{rep}} \geq \varphi_{\text{tot}}^{\text{sub}}$ and $\varphi_{\text{tot}}^{\text{rep}} \geq \varphi_{\text{tot}}^{\text{split}}$. And we conclude that the total payoff to a malicious player by replication is no less than replicating subsets and splitting. □

Lemma 3.4.2.3. *Under payoff allocation using the Banzhaf value, executing one of the three aforementioned attack actions and acting under two identities yields equal payoff.*

Proof. In the market game $G = (N, v)$, a malicious player i with data $D = D_i \cup D_j$, executes one of the following attack actions and act under identities $\{i_1, i_2\}$:

(a) Replication: both i_1 and i_2 hold the union of all data $i_1 = i_2 = D_i \cup D_j$

(b) Subset Replication: $i_1 = D_i \cup D_j$, and the replica holds a subset $i_2 = D_i$

(c) Splitting: each identity holds a subset of the data $i_1 = D_i$, and $i_2 = D_j$.

For clarity of presentation, we denote the players and (data) set operators $C \cup D_i$ by $C + i$.

$$\text{(a) } \varphi_{\text{tot}}^{\text{rep}} = \varphi_{i_1} + \varphi_{i_2} = 2 \sum_{C \subseteq N \setminus \{i\}} w_{|C|,N+1}(v(C+i+j) - v(C))$$

$$\begin{aligned}\text{(b) } \varphi_{\text{tot}}^{\text{sub}} &= \varphi_{i_1} + \varphi_{i_2} \\ &= \sum_{C \subseteq N \setminus \{i\}} w_{|C|,N+1}(v(C+i+j) - v(C)) + \sum_{C \subseteq N \setminus \{i\}} w_{|C+i_2|,N+1}(v(C+i+j) - v(C+i)) + \\ &\quad \sum_{C \subseteq N \setminus \{i\}} w_{|C|,N+1}(v(C+i) - v(C)) \\ &= \sum_{C \subseteq N \setminus \{i\}} (w_{|C|,N+1} + w_{|C|+1,N+1})v(C+i+j) - 2w_{|C|,N+1}v(C) + (w_{|C|,N+1} - w_{|C|+1,N+1})v(C+i)\end{aligned}$$

$$\begin{aligned}\text{(c) } \varphi_{\text{tot}}^{\text{split}} &= \varphi_{i_1} + \varphi_{i_2} \\ &= \sum_{C \subseteq N \setminus \{i\}} [w_{|C|,N+1}(v(C+i) + v(C+j) - 2v(C)) + w_{|C+i_2|,N+1}(v(C+i+j) - v(C+j)) + \\ &\quad w_{|C+i_1|,N+1}(v(C+i+j) - v(C+i))] \\ &= \sum_{C \subseteq N \setminus \{i\}} [w_{|C|,N+1}(v(C+i) + v(C+j) - 2v(C)) + w_{|C|+1,N+1}(2v(C+i+j) - v(C+i) - v(C+j))]\end{aligned}$$

The difference between the total payoff by replication, subset replication and splitting is:

$$\begin{aligned}\varphi_{\text{tot}}^{\text{rep}} - \varphi_{\text{tot}}^{\text{sub}} &= \sum_{C \subseteq N \setminus \{i\}} (w_{|C|,N+1} - w_{|C|+1,N+1})(v(C+i+j) - v(C+i)) \\ \varphi_{\text{tot}}^{\text{sub}} - \varphi_{\text{tot}}^{\text{split}} &= \sum_{C \subseteq N \setminus \{i\}} (w_{|C|,N+1} - w_{|C|+1,N+1})(v(C+i+j) - v(C+j))\end{aligned}$$

Note that $w_{|C|,N+1} = w_{|C|+1,N+1}$ for the Banzhaf value, since the Banzhaf weights are constant by definition. Hence, $\varphi_{\text{tot}}^{\text{rep}} = \varphi_{\text{tot}}^{\text{sub}}$ and $\varphi_{\text{tot}}^{\text{sub}} = \varphi_{\text{tot}}^{\text{split}}$. Therefore, under the Banzhaf value, executing any one of the three attack actions would yield equal payoff. □

A.8 Proofs for Section 3.4.3

Theorem 3.4.3.1. *Let* $\varphi_i = \sum_{c=0}^{|N|-1} \alpha_c z_i(c)$ *denote a semivalue. Algorithm 5 uses unbiased estimators* $\hat{\varphi}_i = \sum_{c=0}^{|N|-1} \alpha_c \frac{|N|}{|N|-c}[\frac{|N|-c}{|N|}\bar{U}_i(c+1) + \frac{c}{|N|}\bar{U}_i(c) - U_c]$ *for the payoff of each player i, and* $\hat{\varphi}_{\text{all}} = N\sum_{c=0}^{|N|-1} \alpha_c(U_{c+1} - U_c)$ *for the total payoff allocated to all players, where* U_c *is the mean of size c coalitions, and* $\bar{U}_i(c)$ *is the mean of size c coalitions which contain player i.*

Proof. We now show that $\hat{\varphi}_i$ and $\hat{\varphi_{\text{all}}}$ are unbiased estimators for φ_i and φ_{all} respectively, i.e., $\mathbb{E}[\hat{\varphi}_i] = \varphi_i$ and $\mathbb{E}[\hat{\varphi_{\text{all}}}] = \varphi_{\text{all}}$.

(1) Proof for $\mathbb{E}[\hat{\varphi}_i] = \varphi_i$**:** Let $[N]^c$ denote size-c subsets: $[N]^c = \{C | C \subseteq N, |C| = c\}$.

$$
\begin{aligned}
\mathbb{E}[\hat{\varphi}_i] &= \sum_{c=0}^{N-1} \alpha_c \tfrac{N}{N-c}\big(\tfrac{N-c}{N}\mathbb{E}[\bar{U}_i(c+1)] + \tfrac{c}{N}\mathbb{E}[\bar{U}_i(c)] - \mathbb{E}[U_c]\big) \\
&= \sum_{c=0}^{N-1} \alpha_c \tfrac{N}{N-c}\big(\tfrac{N-c}{N}\frac{\sum_{S\subseteq[N]^{c+1}, i\in C} v(C)}{\binom{N-1}{c}} + \tfrac{c}{N}\frac{\sum_{C\subseteq[N]^c, i\in C} v(C)}{\binom{N-1}{c-1}} - \frac{\sum_{C\subseteq[N]^c} v(C)}{\binom{N}{c}}\big) \\
&= \sum_{c=0}^{N-1} \alpha_c \tfrac{1}{\binom{N-1}{c}}\big(\sum_{C\subseteq[N]^{c+1}, i\in C} v(C) + \sum_{C\subseteq[N]^c, i\in C} v(C) - \sum_{C\subseteq[N]^c} v(C)\big) \\
&= \sum_{c=0}^{N-1} \alpha_c \tfrac{1}{\binom{N-1}{c}}\big(\sum_{C\subseteq[N]^c, i\notin C} v(C\cup\{i\}) + \sum_{C\subseteq[N]^c, i\in C} v(C) - \sum_{C\subseteq[N]^c} v(C)\big) \\
&= \sum_{c=0}^{N-1} \alpha_c \tfrac{1}{\binom{N-1}{c}} \sum_{C\subseteq[N]^c, i\notin C} \big(v(C\cup\{i\}) - v(C)\big) \\
&= \sum_{c=0}^{N-1} \alpha_c z_i(c) \\
&= \varphi_i
\end{aligned}
$$

(2) Proof for $\mathbb{E}[\hat{\varphi}_{\text{all}}] = \varphi_{\text{all}}$:

$$
\begin{aligned}
\varphi_{\text{all}} &= \sum_{i\in N}\sum_{c=0}^{N-1} \alpha_c z_i(c) \\
&= \sum_{i\in N}\sum_{c=0}^{N-1} \alpha_c \tfrac{1}{\binom{N-1}{c}} \sum_{C\subseteq[N\setminus\{i\}]^c} (v(C\cup\{i\}) - v(C)) \\
&= \sum_{i\in N}\sum_{c=0}^{N-1} \alpha_c \tfrac{1}{\binom{N-1}{c}} \sum_{C\subseteq[N]^c} (v(C\cup\{i\}) - v(S)) \\
&= \sum_{c=0}^{N-1} \alpha_c \tfrac{1}{\binom{N-1}{c}} \sum_{C\subseteq[N]^c} \big([\sum_{i\in N} v(C\cup\{i\})] - N v(C)\big) \\
&= \sum_{c=0}^{N-1} \alpha_c \tfrac{1}{\binom{N-1}{c}} \big(\sum_{C\subseteq[N]^c}\sum_{i\in N} v(C\cup\{i\}) - N\sum_{C\subseteq[N]^c} v(C)\big) \\
&= \sum_{c=0}^{N-1} \alpha_c \tfrac{1}{\binom{N-1}{c}} \big(\sum_{i\in N}\sum_{C\subseteq[N]^c, i\notin C} v(C\cup\{i\}) + \sum_{i\in N}\sum_{C\subseteq[N]^c, i\in C} v(C\cup\{i\}) - N\sum_{C\subseteq[N]^c} v(C)\big) \\
&= \sum_{c=0}^{N-1} \alpha_c \tfrac{1}{\binom{N-1}{c}} \big(\sum_{i\in N}\sum_{C\subseteq[N]^{c+1}, i\in C} v(C) + \sum_{i\in N}\sum_{C\subseteq[N]^c, i\in C} v(C) - N\sum_{C\subseteq[N]^c} v(C)\big) \\
&= \sum_{c=0}^{N-1} \alpha_c \tfrac{1}{\binom{N-1}{c}} \big(\tfrac{N\binom{N-1}{c}}{\binom{N}{c+1}} \sum_{C\subseteq[N]^{c+1}} v(C) + \tfrac{N\binom{N-1}{c-1}}{\binom{N}{c}} \sum_{C\subseteq[N]^c} v(C) - N\sum_{C\subseteq[N]^c} v(C)\big) \\
&= N\sum_{c=0}^{N-1} \alpha_c \big(\tfrac{\sum_{C\subseteq[N]^{c+1}} v(C)}{\binom{N}{c+1}} + (\tfrac{\binom{N-1}{c-1}}{\binom{N}{c}} - 1)\tfrac{\sum_{C\subseteq[N]^c} v(C)}{\binom{N-1}{c}}\big) \\
&= N\sum_{c=0}^{N-1} \alpha_c \big(\tfrac{\sum_{C\subseteq[N]^{c+1}} v(C)}{\binom{N}{c+1}} + \tfrac{c-N}{N}\tfrac{\sum_{C\subseteq[N]^c} v(C)}{\binom{N-1}{c}}\big) \\
&= N\sum_{c=0}^{N-1} \alpha_c \big(\tfrac{\sum_{C\subseteq[N]^{c+1}} v(C)}{\binom{N}{c+1}} + \tfrac{\sum_{C\subseteq[N]^c} v(C)}{\binom{N}{c}}\big) \\
&= N\sum_{c=0}^{N-1} \alpha_c \big(\mathbb{E}[\hat{U}_{c+1}] - \mathbb{E}[\hat{U}_c]\big) \\
&= \mathbb{E}[\hat{\varphi}_{\text{all}}]
\end{aligned}
$$

□

Bibliography

[1] Abbeel, P. and Ng, A. Y. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine learning*, 2004.

[2] Agarwal, A., Dahleh, M., and Sarkar, T. A marketplace for data: An algorithmic solution. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pp. 701–726, 2019.

[3] Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P., and Zaremba, W. Hindsight experience replay. *arXiv preprint arXiv:1707.01495*, 2017.

[4] Aziz, H. and Paterson, M. False name manipulations in weighted voting games: splitting, merging and annexation. *arXiv preprint arXiv:0905.3348*, 2009.

[5] Aziz, H., Bachrach, Y., Elkind, E., and Paterson, M. False-name manipulations in weighted voting games. *Journal of Artificial Intelligence Research*, 40:57–93, 2011.

[6] Bachrach, Y., Markakis, E., Procaccia, A. D., Rosenschein, J. S., and Saberi, A. Approximating power indices. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 943–950, 2008.

[7] Bachrach, Y., Parkes, D. C., and Rosenschein, J. S. Computing cooperative solution concepts in coalitional skill games. *Artificial Intelligence*, 204:1–21, 2013.

[8] Banzhaf III, J. F. Weighted voting doesn't work: A mathematical analysis. *Rutgers L. Rev.*, 19:317, 1964.

[9] Barcelos, H. C., Mendoza, M. R., and Moreira, V. P. Identifying and fusing duplicate features for data mining. In *SBBD*, pp. 133–144, 2020.

[10] Bernstein, D. S., Givan, R., Immerman, N., and Zilberstein, S. The complexity of decentralized control of markov decision processes. *Mathematics of operations research*, 27(4):819–840, 2002.

[11] Bilmes, J. and Bai, W. Deep submodular functions. *arXiv preprint arXiv:1701.08939*, 2017.

[12] Boutilier, C. Planning, learning and coordination in multiagent decision processes. In *TARK*, volume 96, pp. 195–210. Citeseer, 1996.

[13] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[14] Camburu, O.-M. Explaining deep neural networks. *arXiv preprint arXiv:2010.01496*, 2020.

[15] Carreras, F. and Freixas, J. A note on regular semivalues. *International Game Theory Review*, 2(04):345–352, 2000.

[16] Carreras, F. and Giménez, J. M. Semivalues: power, potential and multilinear extensions. 2010.

[17] Carreras, F. and Puente, M. A. Symmetric coalitional binomial semivalues. *Group Decision and Negotiation*, 21(5):637–662, 2012.

[18] Castro, J., Gómez, D., and Tejada, J. Polynomial calculation of the shapley value based on sampling. *Computers & Operations Research*, 36(5):1726–1730, 2009.

[19] Castro, J., Gómez, D., Molina, E., and Tejada, J. Improving polynomial estimation of the shapley value by stratified random sampling with optimum allocation. *Computers & Operations Research*, 82:180–188, 2017.

[20] Chalkiadakis, G. and Boutilier, C. Coordination in multiagent reinforcement learning: A bayesian approach. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pp. 709–716, 2003.

[21] Chalkiadakis, G. and Boutilier, C. Bayesian reinforcement learning for coalition formation under uncertainty. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pp. 1090–1097, 2004.

[22] Chalkiadakis, G., Elkind, E., and Wooldridge, M. *Computational Aspects of Cooperative Game Theory*. Morgan & Claypool Publishers, 2011.

[23] Chen, J., Song, L., Wainwright, M. J., and Jordan, M. I. L-shapley and c-shapley: Efficient model interpretation for structured data. In *International Conference on Learning Representations (ICLR)*, 2019.

[24] Chen, L., Koutris, P., and Kumar, A. Towards model-based pricing for machine learning in a data marketplace. *Proceedings of the 2019 International Conference on Management of Data*, 2019.

[25] Chou, P.-W., Maturana, D., and Scherer, S. Improving stochastic policy gradients in continuous control with deep reinforcement learning using the beta distribution. In *International conference on machine learning*, pp. 834–843. PMLR, 2017.

[26] Chua, K., Calandra, R., McAllister, R., and Levine, S. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *arXiv preprint arXiv:1805.12114*, 2018.

[27] Cohen, S. B., Ruppin, E., and Dror, G. Feature selection based on the shapley value. In *International Joint Conference on Artificial Intelligence (IJCAI)*, volume 5, pp. 665–670, 2005.

[28] Cornuejols, G., Fisher, M., and Nemhauser, G. L. On the uncapacitated location problem. In *Annals of Discrete Mathematics*, volume 1, pp. 163–177. Elsevier, 1977.

[29] Das, A., Dasgupta, A., and Kumar, R. Selecting diverse features via spectral regularization. *Advances in neural information processing systems*, 25:1583–1591, 2012.

[30] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[31] Dua, D. and Graff, C. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

[32] Dubey, P., Neyman, A., and Weber, R. J. Value theory without efficiency. *Mathematics of Operations Research*, 6(1):122–128, 1981.

[33] Durfee, E. H., Lesser, V. R., and Corkill, D. D. Trends in cooperative distributed problem solving. *IEEE Transactions on Knowledge and Data Engineering*, 1:63–83, 1995.

[34] Ebert, F., Finn, C., Dasari, S., Xie, A., Lee, A., and Levine, S. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018.

[35] Fatima, S. S., Wooldridge, M., and Jennings, N. R. A linear approximation method for the shapley value. *Artificial Intelligence*, 172(14):1673–1699, 2008.

[36] Fazeli, N., Oller, M., Wu, J., Wu, Z., Tenenbaum, J. B., and Rodriguez, A. See, feel, act: Hierarchical learning for complex manipulation skills with multisensory fusion. *Science Robotics*, 4(26), 2019.

[37] Feinberg, V., Wan, A., Stoica, I., Jordan, M. I., Gonzalez, J. E., and Levine, S. Model-based value estimation for efficient model-free reinforcement learning. *arXiv preprint arXiv:1803.00101*, 2018.

[38] Finn, C., Levine, S., and Abbeel, P. Guided cost learning: Deep inverse optimal control via policy optimization. In *International conference on machine learning*, pp. 49–58. PMLR, 2016.

[39] Fioretto, F., Pontelli, E., and Yeoh, W. Distributed constraint optimization problems and applications: A survey. *Journal of Artificial Intelligence Research*, 61:623–698, 2018.

[40] Fisac, J. F., Akametalu, A. K., Zeilinger, M. N., Kaynama, S., Gillula, J., and Tomlin, C. J. A general safety framework for learning-based control in uncertain robotic systems. *IEEE Transactions on Automatic Control*, 64(7):2737–2752, 2018.

[41] Fisher, M. L., Nemhauser, G. L., and Wolsey, L. A. An analysis of approximations for maximizing submodular set functions—ii. In *Polyhedral combinatorics*, pp. 73–87. Springer, 1978.

[42] Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., and Whiteson, S. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[43] Foerster, J. N. *Deep multi-agent reinforcement learning*. PhD thesis, University of Oxford, 2018.

[44] Freepik. Graphic resources for everyone, Nov 2021. URL `https://www.freepik.com/`.

[45] Fricker, S. and Maksimov, Y. Pricing of data products in data marketplaces. In *ICSOB*, 2017.

[46] Ghorbani, A. and Zou, J. Data shapley: Equitable valuation of data for machine learning. In *International Conference on Machine Learning (ICML)*, pp. 2242–2251, 2019.

[47] Greenwald, A., Hall, K., and Serrano, R. Correlated q-learning. In *ICML*, volume 3, pp. 242–249, 2003.

[48] Gu, S., Holly, E., Lillicrap, T., and Levine, S. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 3389–3396. IEEE, 2017.

[49] Ha, D. and Schmidhuber, J. World models. *arXiv preprint arXiv:1803.10122*, 2018.

[50] Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. PMLR, 2018.

[51] Haller, H. Collusion properties of values. *International Journal of Game Theory*, 23(3): 261–281, 1994.

[52] Hamrick, J. B., Friesen, A. L., Behbahani, F., Guez, A., Viola, F., Witherspoon, S., Anthony, T., Buesing, L., Veličković, P., and Weber, T. On the role of planning in model-based deep reinforcement learning. *arXiv preprint arXiv:2011.04021*, 2020.

[53] Han, D., Boehmer, W., Wooldridge, M., and Rogers, A. Multi-agent hierarchical reinforcement learning with dynamic termination. In *Pacific Rim International Conference on Artificial Intelligence*, pp. 80–92. Springer, 2019.

[54] Han, D., Wooldridge, M., Rogers, A., Tople, S., Ohrimenko, O., and Tschiatschek, S. Replication-robust payoff-allocation for machine learning data markets. *arXiv preprint arXiv:2006.14583*, 2020.

[55] Han, D., Harrenstein, P., Nugent, S., Philpott, J., and Wooldridge, M. Behavioural strategies in weighted boolean games. *Information and Computation*, 276:104556, 2021.

[56] Han, D., Lu, C. X., Michalak, T., and Wooldridge, M. Multiagent model-based credit assignment for continuous control. *arXiv preprint arXiv:2112.13937*, 2021.

[57] Han, D., Wooldridge, M., and Tschiatschek, S. Mdp abstraction with successor features. *arXiv preprint arXiv:2110.09196*, 2021.

[58] Hill, A., Raffin, A., Ernestus, M., Gleave, A., Kanervisto, A., Traore, R., Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., and Wu, Y. Stable baselines. `https://github.com/hill-a/stable-baselines`, 2018.

[59] Hu, J. and Wellman, M. P. Nash q-learning for general-sum stochastic games. *Journal of machine learning research*, 4(Nov):1039–1069, 2003.

[60] Ichiishi, T. *Game theory for economic analysis*. Elsevier, 2014.

[61] Janner, M., Fu, J., Zhang, M., and Levine, S. When to trust your model: Model-based policy optimization. *arXiv preprint arXiv:1906.08253*, 2019.

[62] Janzing, D., Minorics, L., and Blöbaum, P. Feature relevance quantification in explainable ai: A causal problem. In *International Conference on Artificial Intelligence and Statistics*, pp. 2907–2916. PMLR, 2020.

[63] Jia, R., Dao, D., Wang, B., Hubis, F. A., Hynes, N., Gürel, N. M., Li, B., Zhang, C., Song, D., and Spanos, C. J. Towards efficient data valuation based on the shapley value. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 1167–1176, 2019.

[64] Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[65] Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[66] Kirchhoff, K. and Bilmes, J. Submodularity for data selection in machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 131–141, 2014.

[67] Knudsen, P. H. and Østerdal, L. P. Merging and splitting in cooperative games: some (im) possibility results. *International Journal of Game Theory*, 41(4):763–774, 2012.

[68] Kober, J., Bagnell, J. A., and Peters, J. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[69] Krause, A., Singh, A., and Guestrin, C. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9(2), 2008.

[70] Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, 2009.

[71] LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *nature*, 521(7553):436–444, 2015.

[72] Lehrer, E. An axiomatization of the banzhaf value. *International Journal of Game Theory*, 17(2):89–99, 1988.

[73] Levine, S., Finn, C., Darrell, T., and Abbeel, P. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[74] Li, F., Karpathy, A., and Johnson, J. Tiny imagenet visual recognition challenge. URL `https://tiny-imagenet.herokuapp.com/`.

[75] Li, J., Kuang, K., Wang, B., Liu, F., Chen, L., Wu, F., and Xiao, J. Shapley counterfactual credits for multi-agent reinforcement learning. *arXiv preprint arXiv:2106.00285*, 2021.

[76] Liang, F., Yu, W., An, D., Yang, Q., Fu, X., and Zhao, W. A survey on big data market: Pricing, trading and protection. *IEEE Access*, 6:15132–15154, 2018.

[77] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[78] Lin, H. and Bilmes, J. A class of submodular functions for document summarization. In *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, pp. 510–520, 2011.

[79] Littman, M. L. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pp. 157–163. Elsevier, 1994.

[80] Littman, M. L. Markov games as a framework for multi-agent reinforcement learning. In *Machine Learning Proceedings 1994*, pp. 157–163. Elsevier, 1994.

[81] Liu, Z. and Zhou, J. Introduction to graph neural networks. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 14(2):1–127, 2020.

[82] López, S. and Saboya, M. On the relationship between shapley and owen values. *Central European Journal of Operations Research*, 17(4):415–423, 2009.

[83] Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., and Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. *arXiv preprint arXiv:1706.02275*, 2017.

[84] Lundberg, S. and Lee, S.-I. A unified approach to interpreting model predictions. *arXiv preprint arXiv:1705.07874*, 2017.

[85] Lundberg, S. M., Erion, G. G., and Lee, S.-I. Consistent individualized feature attribution for tree ensembles. *arXiv preprint arXiv:1802.03888*, 2018.

[86] Maleki, S., Tran-Thanh, L., Hines, G., Rahwan, T., and Rogers, A. Bounding the estimation error of sampling-based shapley value approximation. *arXiv preprint arXiv:1306.4265*, 2013.

[87] Mallozzi, P., Pelliccione, P., Knauss, A., Berger, C., and Mohammadiha, N. Autonomous vehicles: State of the art, future trends, and challenges. *Automotive Systems and Software Engineering*, pp. 347–367, 2019.

[88] Michalak, T. P., Aadithya, K. V., Szczepanski, P. L., Ravindran, B., and Jennings, N. R. Efficient computation of the shapley value for game-theoretic network centrality. *Journal of Artificial Intelligence Research*, 46:607–650, 2013.

[89] Mitchell, S., OSullivan, M., and Dunning, I. Pulp: a linear programming toolkit for python. *The University of Auckland, Auckland, New Zealand*, pp. 65, 2011.

[90] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[91] Molnar, C. *Interpretable machine learning*. Lulu. com, 2020.

[92] Muschalle, A., Stahl, F., Löser, A., and Vossen, G. Pricing approaches for data markets. In *BIRTE*, 2012.

[93] Nachum, O., Ahn, M., Ponte, H., Gu, S., and Kumar, V. Multi-agent manipulation via locomotion using hierarchical sim2real. *arXiv preprint arXiv:1908.05224*, 2019.

[94] Nagabandi, A., Kahn, G., Fearing, R. S., and Levine, S. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7559–7566. IEEE, 2018.

[95] Nash, J. F. et al. Equilibrium points in n-person games. *Proceedings of the national academy of sciences*, 36(1):48–49, 1950.

[96] Nemhauser, G. L., Wolsey, L. A., and Fisher, M. L. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14(1):265–294, 1978.

[97] Ng, A. Y., Russell, S. J., et al. Algorithms for inverse reinforcement learning. In *International Conference on Machine Learning*, 2000.

[98] Niu, C., Zheng, Z., Wu, F., Tang, S., Gao, X., and Chen, G. Unlocking the value of privacy: Trading aggregate statistics over private correlated data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2031–2040, 2018.

[99] Nwankpa, C., Ijomah, W., Gachagan, A., and Marshall, S. Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378*, 2018.

[100] Oh, J., Guo, Y., Singh, S., and Lee, H. Self-imitation learning. In *International Conference on Machine Learning*, pp. 3878–3887. PMLR, 2018.

[101] Oh, K.-K., Park, M.-C., and Ahn, H.-S. A survey of multi-agent formation control. *Automatica*, 53:424–440, 2015.

[102] Ohrimenko, O., Tople, S., and Tschiatschek, S. Collaborative machine learning markets with data-replication-robust payments. *arXiv preprint arXiv:1911.09052*, 2019.

[103] Ohta, N., Conitzer, V., Satoh, Y., Iwasaki, A., and Yokoo, M. Anonymity-proof shapley value: extending shapley value for coalitional games in open environments. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2*, pp. 927–934, 2008.

[104] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019.

[105] Pei, J. A survey on data pricing: from economics to data science. *ArXiv*, abs/2009.04462, 2020.

[106] Prianto, E., Kim, M., Park, J.-H., Bae, J.-H., and Kim, J.-S. Path planning for multi-arm manipulators using deep reinforcement learning: Soft actor–critic with hindsight experience replay. *Sensors*, 20(20):5911, 2020.

[107] Rapoport, A., Chammah, A. M., and Orwant, C. J. *Prisoner's dilemma: A study in conflict and cooperation*, volume 165. University of Michigan press, 1965.

[108] Rashid, T., Samvelyan, M., Schroeder, C., Farquhar, G., Foerster, J., and Whiteson, S. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, pp. 4295–4304. PMLR, 2018.

[109] Rashid, T., Farquhar, G., Peng, B., and Whiteson, S. Weighted qmix: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning. *arXiv preprint arXiv:2006.10800*, 2020.

[110] Roh, Y., Heo, G., and Whang, S. E. A survey on data collection for machine learning: a big data-ai integration perspective. *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[111] Roth, A. E. *The Shapley value: essays in honor of Lloyd S. Shapley*. Cambridge University Press, 1988.

[112] Sadigh, D., Landolfi, N., Sastry, S. S., Seshia, S. A., and Dragan, A. D. Planning for cars that coordinate with people: leveraging effects on human actions for planning and active information gathering over human internal state. *Autonomous Robots*, 42(7):1405–1426, 2018.

[113] Salhi, S. Discrete location theory. *Journal of the Operational Research Society*, 42: 1124–1125, 1991.

[114] Samvelyan, M., Rashid, T., De Witt, C. S., Farquhar, G., Nardelli, N., Rudner, T. G., Hung, C.-M., Torr, P. H., Foerster, J., and Whiteson, S. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

[115] Schomm, F., Stahl, F., and Vossen, G. Marketplaces for data: an initial survey. *SIGMOD Rec.*, 42:15–26, 2013.

[116] Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015.

[117] Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[118] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[119] Sen, S., Joe-Wong, C., Ha, S., and Chiang, M. A survey of smart data pricing: Past proposals, current plans, and future trends. *Acm computing surveys (csur)*, 46(2):1–37, 2013.

[120] Shapley, L. S. A value for n-person games. *Contributions to the Theory of Games*, 2(28): 307–317, 1953.

[121] Shapley, L. S. Cores of convex games. *International journal of game theory*, 1(1):11–26, 1971.

[122] Shapley, L. S. *17. A Value for n-Person Games*, pp. 307–318. Princeton University Press, 2016. doi: doi:10.1515/9781400881970-018. URL https://doi.org/10.1515/9781400881970-018.

[123] Shoham, Y. and Leyton-Brown, K. *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press, 2008.

[124] Shorten, C. and Khoshgoftaar, T. M. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

[125] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[126] Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[127] Spong, M. W., Hutchinson, S., Vidyasagar, M., et al. *Robot modeling and control*, volume 3. wiley New York, 2006.

[128] Stadelmann, T., Amirian, M., Arabaci, I., Arnold, M., Duivesteijn, G. F., Elezi, I., Geiger, M., Lörwald, S., Meier, B. B., Rombach, K., et al. Deep learning in the wild. In *IAPR Workshop on Artificial Neural Networks in Pattern Recognition*, pp. 17–38. Springer, 2018.

[129] Stone, P. and Veloso, M. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, 2000.

[130] Štrumbelj, E. and Kononenko, I. Explaining prediction models and individual predictions with feature contributions. *Knowledge and information systems*, 41(3):647–665, 2014.

[131] Sundararajan, M. and Najmi, A. The many shapley values for model explanation. In *International Conference on Machine Learning*, pp. 9269–9278. PMLR, 2020.

[132] Sutton, R. S. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

[133] Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

[134] Sutton, R. S., Precup, D., and Singh, S. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[135] Syed, U., Bowling, M., and Schapire, R. E. Apprenticeship learning using linear programming. In *International Conference on Machine Learning*, pp. 1032–1039, 2008.

[136] Szczepański, P. L., Tarkowski, M. K., Michalak, T. P., Harrenstein, P., and Wooldridge, M. Efficient computation of semivalues for game-theoretic network centrality. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[137] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[138] Tan, M. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pp. 330–337, 1993.

[139] Tan, M. Readings in agents. chapter Multi-agent Reinforcement Learning: Independent vs. Cooperative Agents, pp. 487–494. 1998.

[140] Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033. IEEE, 2012.

[141] van den Brink, R. Efficiency and collusion neutrality in cooperative games and networks. *Games and Economic Behavior*, 76(1):344–348, 2012.

[142] Wang, J., Ren, Z., Liu, T., Yu, Y., and Zhang, C. Qplex: Duplex dueling multi-agent q-learning. *arXiv preprint arXiv:2008.01062*, 2020.

[143] Wang, J., Zhang, Y., Kim, T.-K., and Gu, Y. Shapley q-value: a local reward approach to solve global reward games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7285–7292, 2020.

[144] Wang, J., Wang, J., Zhang, Y., Gu, Y., and Kim, T.-K. Shaq: Incorporating shapley value theory into q-learning for multi-agent reinforcement learning. *arXiv preprint arXiv:2105.15013*, 2021.

[145] Wang, T., Liao, R., Ba, J., and Fidler, S. Nervenet: Learning structured policy with graph neural networks. In *International Conference on Learning Representations*, 2018.

[146] Weng, L. A long peak into reinforcement learning. URL `https://lilianweng.github.io/lil-log/contact.html`.

[147] Wolpert, D. H. and Tumer, K. Optimal payoff functions for members of collectives. In *Modeling complexity in economic and social systems*, pp. 355–369. World Scientific, 2002.

[148] Wooldridge, M. *An introduction to multiagent systems*. John wiley & sons, 2009.

[149] Xu, J., Zhong, F., and Wang, Y. Learning multi-agent coordination for enhancing target coverage in directional sensor networks. *arXiv preprint arXiv:2010.13110*, 2020.

[150] Yu, C., Velu, A., Vinitsky, E., Wang, Y., Bayen, A., and Wu, Y. The surprising effectiveness of mappo in cooperative, multi-agent games. *arXiv preprint arXiv:2103.01955*, 2021.

[151] Yu, H. and Zhang, M. Data pricing strategy based on data quality. *Comput. Ind. Eng.*, 112: 1–10, 2017.

[152] Zhang, M. and Beltrán, F. A survey of data pricing methods. *Available at SSRN 3609120*, 2020.

[153] Zhang, Z., Cui, P., and Zhu, W. Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[154] Zhou, L., Tzoumas, V., Pappas, G. J., and Tokekar, P. Distributed attack-robust submodular maximization for multi-robot planning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2479–2485. IEEE, 2020.

[155] Ziebart, B. D., Maas, A. L., Bagnell, J. A., and Dey, A. K. Maximum entropy inverse reinforcement learning. In *AAAI Conference on Artificial Intelligence*, pp. 1433–1438, 2008.

[156] Zoabi, Y., Deri-Rozov, S., and Shomron, N. Machine learning-based prediction of covid-19 diagnosis based on symptoms. *npj digital medicine*, 4(1):1–5, 2021.