UNIVERSITY OF SOUTHAMPTON

FACULTY OF BUSINESS, LAW AND ART

Southampton Business School

Payoff Allocation Methods for Several Operational Research Games

by

Phuoc Hoang Le

Thesis for the degree of Doctor of Philosophy

May 2017

University of Southampton

Doctoral Thesis

Payoff Allocation Methods for Several Operational Research Games

Author: Phuoc Hoang Le

Supervisor:
Dr. Tri-Dung Nguyen
Prof. Tolga Bektaş

Examiners:
Dr. Patrick Beullens
Dr. Katerina Papadaki

Southampton 2016

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF BUSINESS, LAW AND ART
Southampton Business School

Doctor of Philosophy in Management Science

PAYOFF ALLOCATION METHODS FOR SEVERAL OPERATIONAL RESEARCH GAMES

by Phuoc Hoang Le

From the start of this century, enforcing and maintaining collaborations became one of the trends in business and management, ranging from production and inventory to transportation and network communication. Because of the economies of scale, many enterprises might collaborate to generate more value and save costs instead of working on their own. The rapid change in science and information technologies enables users to communicate and share data more efficiently through different digital platforms. These emerging technologies open several opportunities for companies and institutions to allow potential coordination and collaboration. This development has created the sharing economy which enables individuals, households, businesses, and organisations to engage in collaborative production, distribution, and consumption of goods and services.

The interactions among players willing to cooperate are the underlying theme of Cooperative Game Theory. In this field, payoff (yield/cost) sharing problems aim at producing solutions with two main criteria: fairness and stability. These principles lead to some popular solution concepts such as the Shapley value and the core, which concern distributing the payoff to members of a coalition. The main contributions of this thesis are the developments of some computationally efficient methods to calculate these solution concepts in the generalised minimum spanning tree, linear production, and multidimensional integer knapsack games. These are the subclasses of operational research games where the characteristic functions of these cooperative games are described by linear or integer linear programming formulations.

The minimum-cost spanning tree game is a particular class of cooperative games defined on a graph, where each player owns a vertex. Solutions of the game represent ways to distribute the total cost of a minimum-cost spanning tree among all the players. When we partition the graph into clusters, the generalised minimum spanning tree problem is to determine a minimum-cost tree including exactly one vertex from each cluster.

The first chapter introduces and studies the generalised minimum spanning tree game and some of its properties. We propose a constraint generation algorithm to calculate a stable payoff distribution and present some computational results obtained using the proposed algorithm.

Another class of operational research games, called linear production game, is concerned with allocating the total payoff of an enterprise among the owners of the resources in a fair way. With cooperative game theory provides a mathematical framework for sharing the benefit of the cooperation, the Shapley value is one of the widely used solution concepts as a fair measurement in this area. Finding the exact Shapley values for linear production games is, however, challenging when the number of players exceeds 30. This chapter describes the deploy of linear programming sensitivity analysis for more efficient computation. We also propose the stratified sampling technique to estimate the Shapley value for large-scale linear production games. Computational results show the effectiveness of these compared to other existing methods.

The integer optimisation game, where the characteristic function is generated by solving an integer optimisation problem, forms an important class in cooperative game theory. In this chapter, we introduce a new subclass of integer optimisation games namely the multidimensional integer knapsack game. Finding the Shapley value is challenging for most integer optimisation games with a large number of players, particularly for the multidimensional integer knapsack game due to the structure of its characteristic function. We describe an algebraic approach using Gröbner bases to be able to compute the Shapley value efficiently. Some computational experiments are presented to show the potential of the proposed method compared to CPLEX, the state-of-the-art commercial solver on some randomly generated instances.

Tóm tắt khóa luận

PHƯƠNG PHÁP PHÂN PHỐI KẾT QUẢ CHO MỘT SỐ TRÒ CHƠI VẬN TRÙ

Từ bắt đầu của thế kỷ 21, việc tăng cường và duy trì sự cộng tác đã trở thành một xu hướng mạnh trong kinh doanh và quản lý, điều đó được thể hiện trong nhiều lĩnh vực từ sản xuất và lưu trữ hàng hoá cho đến vận chuyển và mạng lưới thông tin. Do lợi thế kinh tế nhờ quy mô, các cá thể thường muốn hợp tác để tạo ra nhiều giá trị hơn và tiết kiệm chi phí thay vì hoạt động riêng lẻ. Sự thay đổi nhanh chóng trong khoa học và công nghệ thông tin ngày nay cho phép người dùng có thể giao tiếp và chia sẻ dữ liệu hiệu quả thông qua nhiều nền tảng kỹ thuật số khác nhau. Những công nghệ mới đó đã mở ra nhiều cơ hội hợp tác cho các công ty và tập đoàn kinh doanh. Sự phát triển này tạo ra nền kinh tế chia sẻ để cho phép các cá nhân, hộ gia đình, và công ty có khả năng cùng sản xuất, phân phối và tiêu thụ các hàng hóa và dich vụ.

Sự tương tác giữa những người chơi muốn cùng nhau làm việc là một trong những chủ đề chính của lý thuyết trò chơi hợp tác. Trong ngành nghiên cứu này, sự chia sẻ kết quả (chi phí) nhằm mục đích tìm ra cách phân phối thoả mãn các thành viên tham gia theo hai tiêu chí chính: tính công bằng và tính ổn định. Những nguyên lý này dẫn đến một số khái niệm nổi tiếng trong lý thuyết trò chơi như tập lõi (hạt nhân) và giá trị Shapley, nhằm phân phối kết quả đầu ra cho các thành viên trong nhóm cộng tác. Đóng góp chính của khóa luận là sự phát triển những phương pháp khác nhau để tính toán những khái niệm này trong bài toán nối cây tối thiểu tổng quát, bài toán sản xuất tuyến tính và bài toán balô nhiều chiều. Đây là những lớp con của trò chơi vận trù, trong đó hàm đặc trưng của những trò chơi hợp tác này được mô tả bởi hệ qui hoạch tuyến tính hoặc hệ qui hoạch nguyên.

Trò chơi nối cây tối thiểu là một lớp con của trò chơi hợp tác được định nghĩa trên một đồ thị với tập đỉnh và tập cạnh xác định, trong đó mỗi người chơi sở hữu một đỉnh. Một lời giải của trò chơi này đại diện cho cách phân phối tổng chi phí của bài toán cây tối thiểu cho những thành viên tham gia. Khi chúng ta chia đồ thị này thành những cụm nhỏ, bài toán cây nối tối thiểu tổng quát nhằm mục đích xác định một cây tối thiểu sao cho nó chứa chính xác một đỉnh trong mỗi cụm. Chương đầu tiên này giới thiệu và tìm hiểu trò chơi nối cây tối thiểu và những đặc tính của nó. Chúng tôi đề xuất phương pháp giải thuật tạo lớp cắt nhằm phân phối chi phí một cách ổn định và đưa ra những kết quả tính toán cho phương pháp này.

Trò chơi sản xuất tuyến tính nhằm phân phối một cách công bằng tổng giá trị của kết quả đầu ra của một nhóm đầu tư cho các thành viên sở hữu tài nguyên. Giá trị Shapley

là một trong những khái niệm được dùng rất phổ biến trong lý thuyết trò chơi hợp tác, nhằm cung cấp một khuôn khổ tính toán cho việc chia sẻ kết quả của sự cộng tác một cách công bằng. Tuy nhiên, việc tìm chính xác giá trị Shapley cho trò chơi sản xuất tuyến tính là rất khó khăn khi trò chơi có hơn 30 thành viên. Chương này mô tả cách dùng sự kết hợp giữa phương pháp phân tích độ nhạy qui hoạch tuyến tính và phương pháp lấy mẫu thử xếp tầng để xấp xỉ giá trị Shapley cho trò chơi sản xuất tuyến tính với nhiều thành viên. Kết quả tính toán sẽ chỉ ra sự hiệu quả của những phương pháp này khi chúng tôi so sánh với các cách thức chia sẻ đã có cho cùng bài toán.

Trò chơi tối ưu nguyên, khi hàm đặc trưng là kết quả của một bài toán qui hoạch nguyên, là một trong những lớp quan trọng của lý thuyết trò chơi hợp tác. Trong chương này, chúng tôi giới thiệu một lớp trò chơi tối ưu nguyên mới là trò chơi balô nguyên nhiều chiều. Việc tìm giá trị Shapley cho lớp trò chơi này là khá khó khăn vì cấu trúc của hàm đặc trưng của nó. Chúng tôi sẽ mô tả một phương pháp đại số dùng cơ sở Grobner, nhằm tính toán giá trị Shapley một cách hiệu quả. Một số thử nghiệm tính toán được thực hiện để chỉ ra khả năng của phương pháp này khi so sánh với phần mềm CPLEX trong một số trường hợp được tạo ngẫu nhiên.

Contents

List of Figures

List of Tables

Declaration of Authorship

I, Phuoc Hoang Le , declare that the thesis entitled '*Payoff Allocation Methods for Several Operational Research Games*' and the work presented in it are my own and has been generated by me as the result of my own original research. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University;
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- Where I have consulted the published work of others, this is always clearly attributed;
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- Parts of this work have been published and presented as:
  - *Generalized minimum spanning tree games.* Phuoc Hoang Le, Tri-Dung Nguyen, Tolga Bektaş. EURO Journal on Computational Optimization. **4**, 167–188, 2016.
  - *Efficient computation of the Shapley value for Large-Scale Linear Production Games.* Phuoc Hoang Le, Tri-Dung Nguyen, Tolga Bektaş. European conference for Operational Research, Glasgow, UK 2015.
  - *An algebraic approach to the Shapley value computation in Multidimensional Integer Knapsack Games.* Phuoc Hoang Le, Tri-Dung Nguyen, Tolga Bektaş. International symposium on Combinatorial Optimisation, Canterbury, UK 2016.

Signed: **Phước Hoàng Lê**

Date: **December 2016**

Acknowledgements

This thesis would not have been possible without the guidance, support, and encouragement of many individuals, to whom I would like to express my appreciation here.

First and foremost, I would like to express my sincerest gratitude to my advisors, Dr Tri-Dung Nguyen and Prof. Tolga Bektas, for giving me the possibility to write this thesis with their supervisions. Dr Tri-Dung Nguyen has introduced me to various topics of operational research and especially to cooperative game theory. I would like to thank him for the great example of an academic researcher and especially for his outstanding supports. My appreciation goes to Prof. Tolga Bektas, who has taught me to look at the big picture and always consider other perspectives. His intrigued questions and critical insights impress me over and over again.

My sincere thanks also go to Dr Patrick Beullens and Dr Katerina Papadaki for generously making the time in their busy schedules to serve on my thesis committee. I am grateful to Ms Vesna Perisic for giving me the teaching opportunity in the FY program. Also, thanks to Prof. Chris Potts for having some useful discussions and sharing insights of Management Science topic with me on several occasions.

I would like to thank the professors and colleagues of the CORMSIS group, which seminars have broadened my knowledge to many different topics in Management Science. I would like to acknowledge the Southampton Business School for funding this PhD project. Thank also goes to the SBS administrative staff for providing lots of great help and the Doctoral College for offering many high-quality resources and training for our postgraduate researchers.

In these years, SBS-PGR Society has been a unique home for me as we shared our experience, created many interesting social events, had some fun time together. I am grateful to everyone at SBS-PGR Society, especially my PhD colleagues and friends including Carlos, Ranga, Sterling, Sarah, Murat, Yi, Dave, Jun, Layla, Stefanos, Rahma, Koç, etc. for creating such an enjoyable working environment.

My friends outside the SBS have also been of great supports these past four years. Special thanks to the Vietnamese Soton community for always being there with my family during the craziest of times. We have such a wonderful group of people which always support each other when we need assistances the most. I always hope to be a part of the community in the future.

I would like to sincerely thank my father, my mother and my sister for their encouragements and for always believing in me. My father is my first Mathematics teacher, and he always shows me by example many profound lessons in life. My mother has always

been a great inspiration for me with her hard-working manner and her determination to overcome any challenges to accomplish the goal. Many of my successes today are the outcomes of their guidance, vision, and caring.

Finally, I would like to thank my wife Anh-Thoa and my son Ryan for their endless love and incredible encouragement.

This dissertation is dedicated to my parents

Kính tặng ba mẹ

Abbreviations

MSTP	**M**inimum-cost **S**panning **T**ree **P**roblem
MSTG	**M**inimum-cost **S**panning **T**ree **G**ame
GMSTP	**G**eneralised **M**inimum **S**panning **T**ree **P**roblem
GMSTG	**G**eneralised **M**inimum **S**panning **T**ree **G**ame
OTG	**O**ptimal **T**ree **G**ame
ILP	**I**nteger **L**inear **P**rogram
MILP	**M**ixed **I**nteger **L**inear **P**rogram
LPG	**L**inear **P**roduction **G**ame
LPSA	**L**inear **P**rogramming **S**ensitivity **A**nalysis
MAPE	**M**ean **A**bsolute **P**ercentage **E**rror
RMSE	**R**oot **M**ean **S**quared **E**rror
IOG	**I**nteger **O**ptimisation **G**ame
COG	**C**ombinatorial **O**ptimisation **G**ame
ORG	**O**perational **R**esearch **G**ame
MIKG	**M**ultidimensional **I**nteger **K**napsack **G**ame
MIKP	**M**ultidimensional **I**nteger **K**napsack **P**roblem

Chapter 1

Introduction

In this thesis, we study a variety of cooperative games with operational research structures. Most operational research problems are optimisation problems with only one system designer. However, the game theory framework applies when there are at least two players in the system. We presume that when a group of players works in a coalition, all resources of the group are pulled together so that the group can make the decision as a single decision maker. Therefore, the essential feature of these operational research games is the optimisation aspect where each coalition tries to maximise the payoff of its group. We now introduce the research topic of operational research games.

1.1 Game Theory

Past decades have seen significant interest in mathematical models of conflict and cooperation between rational decision makers by the Operational Research and Management Science communities. The field of Game Theory was originally a branch of microeconomics studying strategic behaviour in competitive situations. The research area was initially developed by Von Neumann & Morgenstern (1944) in their classic book entitled 'Theory of Games and Economic Behavior'. Based on a theory of games of strategy, this work established a revolutionary mathematical theory of economic behaviours, including that of firms, markets, and consumers. Later on, game theory developed to many fruitful areas such as non-cooperative equilibrium (Nash 1950a), cooperative games (Shubik 1962), and auctions (Vickrey 1961). The use of game theory in the field of social sciences has grown and further extended to political, sociological, and psychological applications (see, for example, Maschler et al. 2013).

Game theory could be categorised into non-cooperative and cooperative games. Cooperative games differ from non-cooperative games in that binding agreements are possible

before the game is played. This agreement provides a strong incentive for players to work together to receive the largest total pay-off. Moreover, the two branches of game theory differ in how one formalises the mathematical models. In their definitions, the former is concerned with the specific actions of the players, while the latter focuses on the outcome of collaboration.

The non-cooperative game theory provides a mathematical language and develops useful tools to analyse strategic situations. Its focus is to find the best strategy for each player to optimise their benefits. In the literature, there is a rich body of research on the equilibrium situation such as Nash equilibrium. If it exists, the process helps identify the stable status of a complex system when many players interact with each other. The non-cooperative game theory develops our understanding of many practical areas including zero-sum games in Duopoly market (Gillies 1959), Nash equilibria in determining the expected flow of traffic in a network (Braess et al. 2005), and the perfect equilibrium in a bargaining model (Rubinstein 1982). Although non-cooperative game theory had enjoyed more attention in economic literature than cooperative game theory, the latter is becoming more popular recently.

Cooperative game theory investigates forming productive collaborations and sharing the payoff. In practice, large firms in the market have a tendency to collaborate to distribute resources and tasks, and political parties join forces to build a winning coalition government. There are many cooperative examples of community organisations such as trade unions, non-profit groups, religious groups, community interest groups and neighbourhood task forces. They tend to work together and focus on positive actions to improve conditions in schools, community or other local institutions. In the situations when a coalition can generate some yields or share the costs, fair distribution is critical to maintaining a stable coalition such that no sub-group of players has an incentive to deviate. This helps maintaining the alliance for future collaboration.

There are a few efforts to incorporate and connect the gap between non-cooperative game theory and cooperative game theory into one stream such as the Nash program (see, for example, Nash 1953, Herrero 1989). In the program, one of its main contributions is the Nash bargaining solution, where it tries to support solutions of cooperative NTU-games by the equilibria of non-cooperative games in the strategic form. Many results can be found on this topic, and we refer the reader to Serrano (2005) for a detailed survey.

One does not have to consider the separate actions of the players in cooperative games. These actions are instead abstracted in a characteristic function of the game model. When it is difficult to have the information about the player's moves or the set of actions with its resulting payoffs, cooperative game models are preferable in many complex multi-player systems. Therefore, we utilise cooperative game theory to model the

outcome of some complex business processes and propose methods to compute some solutions in these game models.

1.2 Cooperative Game Theory

The prevalence of cooperation, sharing, and competition in many human activities has made cooperative game theory a fundamental modelling approach in operations research and economics. Cooperative game theory is employed for the situation in which the players, rather than competing, can cooperate by forming independent coalitions. It is concerned with the decision-making process when several decision-makers are involved. Since groups of players operate on a cooperative basis, the game may become a competition between coalitions of players, rather than between individual players. However, in this thesis, we will focus on cooperative games where each coalition's payoff only depends on the actions taken by the players in that coalition, but not on the actions of other parties. The value of an alliance is what it can achieve by coordinating their efforts. Therefore, in this situation, cooperative games are similar to non-cooperative games when one considers each group of players as a working unit.

In real-life situations, cooperative game theory has found many useful applications. These range from economics and business (e.g. for setting insurance premium Lemaire 1991) and for setting interchange fees for ATM bank networks (Gow & Thomas 1998), in law and political science (Bilbao et al. 2002, for computing voting power), and engineering (Chalkiadakis et al. 2011, for coalition structure formation in multi-agent systems). One of the reasons the description of these system fits more into cooperative games is that the model structure is incompatible to the extensive form of non-cooperative games.

An important aspect of cooperative games, which also differentiates it from non-cooperative games, is the feasibility of building coalition of players. There are two parts of the process allowing possible collaboration among players. The first part is the creation of binding agreements about how they play before the actual game begins. It is the common in practice for companies to use contracts and other legal arrangements to make binding agreements with others. The second part of collaboration is whether one can make agreements concerning share of payoff, or transferring side payments from one player to another. This possibility to share payoff or transfer side payment can help some players attracting other players to coordinate their strategy. The aim of this coordination is to increase the outcomes of these players compared to their individual payoffs.

In most theories and examples presented in this thesis, players are not only allowed to make side payments and share payoffs, but they may also arrange binding agreements in

advance. For this cooperation to be possible, we assume the payoffs are in some transferable commodities, such as money. All payoffs are measured in the same exchanged units, and they are not given to individual players, but only to coalitions. Afterwards, the group transfers the total outcome to each player following the agreements at the beginning. Moreover, we assume that there is a mechanism for forming binding agreements, which will not be investigated further. More detail on the concepts, models and terminology of cooperative game theory can be found in section A.1 of the Appendix.

One of the central questions in cooperative game theory is how to share the cost or allocate the total revenue in a fair way so that there is no incentive for any sub-group to break away and form its own coalitions. A few examples of cooperative game theory to allocate the payoff such as the assessment of landing fees in airports (Littlechild & Thompson 1977), the measurement of power in political assemblies (Banzhaf 1965), and the allotment of water among agricultural communities (Dinar et al. 1992). Later on, we focus on transferable utility cooperative games and their popular solution concepts, especially the core and the Shapley value.

In the context of cooperative game theory, the questions about how the players should form alliances and when the players should work alone arise naturally. The answers to these questions allow the creation of coherent groups of players in order to efficiently achieve their collective goals. This popular type of problems is called coalition formation problem. The coalition formation problem has been extensively studied in the game theory and economics literature. Moreover, in computer science and artificial intelligence, the coalition formation problem is becoming a key topic in multi-agent systems (Kraus 1997). However, forming effective coalitions is still a research challenge in the field of cooperative games. We recommend the interested reader to look at the books of Kahan & Rapoport (1984) and Chalkiadakis et al. (2011) for more information about the coalition formation problem. Most of the games we focus on are super-additive games, i.e. there are incentives for the members in the game to collaborate and create the grand coalition for the benefit of all players.

In this thesis, we concentrate on a particular class of cooperative games, called operational research games. The operational research game is constructed by a set of players and its characteristic function, which provides all possible coalition values of the game. For each coalition of these games, its value will be defined as the objective value of an optimisation problem in operational research. These operational research games constitute an important part of cooperative game theory.

1.3 Operational Research Games

Operational Research(OR) and Management Science(MS) have been well-established as one of the central parts of business and management. There are many examples where OR/MS develop analytical models, concepts in helping to illuminate management issues and solve problems in public or private organisations. Applications of OR/MS are numerous in industry such as airlines, manufacturing companies, service organizations, military branches, and in government. Employing techniques from mathematical modelling, statistical analysis, and mathematical optimisation, OR/MS method arrives at optimal or near-optimal solutions to complex decision-making problems. Researchers faced with a new problem must determine which of these techniques are most appropriate given the goals for improvement, the nature of the system, and the constraints on time and computing power.

Recently, cooperative game theory has been applied as mathematical models for understanding the ways companies, and individuals can collaborate, and the operational research approaches are utilised to find the solutions for these games. In the book by Curiel (1997), the author presents a detailed analysis of cooperative games related to some common operational research problems. Borm et al. (2001) also published a survey on the topic of operational research games. Formally, *operational research game* is a cooperative game where coalition payoffs are the outcome of operational research problems such as the linear program, integer program, or a mixed integer linear program.

One typical example of the operational research games is the case when we have a set of resource owners. These players collaborate to create some products (services) and sell at the market prices. The process of transforming resources to final products and selling to the market can be represented by a mathematical programming problem. In the game, the players do not only care about the way to maximise their group result, but they are also interested in the best method to divide the payoff. Since the collaboration usually helps the players achieve better outcome for the group, finding a stable and fair way to allocate the payoff is one of the main questions for this research direction.

Operational research games (ORG) span a large part of cooperative game theory. It ranges from linear production games, combinatorial optimisation games, integer optimisation games; to inventory games, sequencing games, and stochastic cooperative games. The reader can find some interesting works on the applications of game theory applied to operational research and management science problems such as Chatterjee & Samuelson (2001) and Curiel (1997). Among many applications of ORG, we concentrate in this thesis on three main classes of operational research namely Logistics and Supply Chain Management(SCM), Networks problems, and Production Planning.

Logistics and Supply Chain Management The game theory, especially its non-cooperative part, has appeared in many papers and books on the topic. It has been used to solve the problem of producing the appropriate goods or services in the right quality and quantity, and to distribute them efficiently. One can roughly divide the supply chain system into a centralised or decentralised one based on the way it makes the final decision. In the former case, a single decision maker determines the optimal solution that improves the performance of the supply chains, and all players are willing to follow that decision. In the latter system, each player is an independent decision maker with its own incentive. In this situation, supply chain members can either compete or coordinate for their benefits. If the group works together and agrees to a set of strategies, they can enhance the global performance of the supply chain next to their individual payoffs.

In a cooperative situation, it is important that the group divides the benefits and costs among the supply chain members in such a way that each member is satisfied. Otherwise, at least one of the members would not be prepared to form a coalition. This issue brings a lot of potentials for cooperative game theory to apply into supply logistics and SCM. Moreover, with the development of new technology and the Internet, there is an increasing requirement to understand the interaction among independent players within and across firms to build collaborative relationships for shared benefit. Recently, some researchers apply cooperative game theory for the operational research models in supply chain networks (see, for example, Nagarajan & Sosić 2008, Leng & Parlar 2009, Fiestras-Janeiro et al. 2011).

Communication and Social Networks Another successful utilisation of cooperative game theory is for network problems in communication systems and social structures. Cooperation has emerged as a new communication standard to support the need for self-organizing, decentralized, and autonomic systems. Hence, it becomes critical for companies to seek suitable game theoretical tools that allow them to analyse the behaviour and interactions of the nodes in the communication networks. Social networks are another interesting application of cooperative game theory. Nowadays, there are growing numbers of social structures made up of individuals or organisations that are tied by some specific types of interactions, such as business partnerships, information communicating, trade agreements and political alliances. Cooperative game theory builds the models to form good coalitions and facilitate relationships for completing these tasks.

Mathur et al. (2008) apply cooperative game theory to studying the cooperation possibilities between single antenna receivers and transmitters in a Gaussian interference

channel. The authors consider the interaction between the receivers under the two game-models. The first one creates a transferable utility model where the receivers communicate through noise-free channels and jointly decode the received signals. The second model is a non-transferable utility where the receivers cooperate by forming a linear multi-user detector to process their matched filter signals. Moreover, assuming that the receivers form a grand coalition, the authors investigate the transmitters' cooperation problem under the perfect collaboration.

With the coming of the digital age, there are many popular examples for web-based social networks including Facebook, Google+, LinkedIn, etc. These networks are current vehicles to disseminate information on personal news, job openings, and new products. In these social networks, we can build a graph model of the underlying structure using raw data. A node represents an individual in the model, and two nodes are connected if there exists a social interaction between them. Analysing the structure of complex relationships that exist among the nodes in a social network is useful for both service users and service providers. In particular, the collaborations and group structures in real-world are also recorded and modelled in this network graph, hence, we can learn how they emerge and analysing their economic outcomes.

Within the context of social networks, the main topics of cooperative game theory are the measurement of centrality and power as well as allocation rules such as the Myerson value and the Shapley value. Gomezb et al. (2003) propose some centrality measures for social networks based on game theoretical concepts. A cooperative game is considered from the underlying graph to reflect the interactions among individuals in a group of the network. In this game, the Shapley value is proposed to measure centrality because it represents the average marginal contribution made by each node to every possible combination of the other nodes. In particular, the author considers the Shapley value in the game as the power of players.

One of many interesting problems is to rank the nodes utilising the Shapley value and discover the most influential nodes in social networks (Narayanam & Narahari 2010). Aadithya et al. (2010) also define node centrality when both weighted and unweighted networks are considered. Given a graph network and a cooperative game set over it, four types of coalition values are given by a closed-form expression on the network. The authors develop an exact analytical Shapley value formula and present some polynomial time algorithms for computing these node centralities.

Production Planning Production planning problems present the next application class of operational research games. It is critical to integrate the production planning and control systems to any manufacturing unit of organizations to increase the efficiency

and effectiveness. Production planning is required for many business processes such as scheduling, inspection, dispatch, quality management, inventory management, and supply management. Business has applied the tools of game theory so far mainly in the area of strategic and tactical planning. But recently there is an increasingly interest in using cooperative game theory in the field of production planning at the operational level.

In linear production game, an economic production model is constructed, in which players share their resources to produce finished goods. Molina & Tejada (2004) study an extension of the linear production game with committee control to model the situation when the number of players increases uniformly. They also investigate the convergence of the core of this linear production game model to a set of competitive equilibria. Recently, Guardiola et al. (2009) introduce a new class of cooperative games that arises from the cooperation of several companies in a dynamic production-inventory situation. The players have to deal with discrete demand, allowed shortages, and finite planning horizon. In the model, cooperation among players is given by sharing production processes and warehouse facilities. The authors show that the game is totally balanced and the Owen set reduces to a Owen point allocation, in which every firm has to pay the minimum cost of operation.

There are several operational decisions which some groups of collaborators may apply the cooperative game model to investigate such as deciding the right amount to invest in each time period (Weingartner 1966), choosing suitable projects to finance (Peterson 1967), and finding how many products to make for the current market demand (Owen 1975). In this thesis, we investigate those typical questions which the team have to deal with to maximise the payoff and satisfy its members at the same time.

1.4 Research Objectives

In this thesis, we concentrate on operational research games with some particular linear/integer programming structures such as the generalised minimum cost spanning tree game (GMSTG) arising in the energy and telecommunication industry, the linear production game (LPG) emerging in manufacturing and network design, and multidimensional integer knapsack game (MIKG) for capital budgeting and project selection. All of these games are super-additive games, i.e., the players in this games have the incentive to collaborate as a grand coalition instead of working in smaller groups. We will give more detail in each specific chapters about these games and the related allocation problems.

The operational research games have predefined characteristic functions, where each function maps a coalition into a real number, representing the cost or yield of the group when they work together. An implicit assumption for the characteristic function is that one divides the coalition value amongst the members of that coalition in any way that the members choose. We also suppose that the resources and capabilities of players of a coalition is represented only by the value of that coalition; and the value of a coalition depends only on the members of that coalition and not on the members of other coalitions.

In cooperative games with transferable utilities, players can benefit by cooperating, and they can also arrange binding agreements in advance. After the game, payoffs are given to the group and then divided among its members. Therefore, the main aim of the dissertation is following:

"To design, implement and evaluate effective methods for sharing the cost or distributing the reward among players of a cooperative game, where the characteristic function is expressed in the form of a linear or an integer linear programme."

We generate three research objectives for this thesis to achieve this aim:

- To define the cooperative games whose corresponding characteristic functions have some special operational research structures.
- To evaluate a stable payoff allocation in the core or least core for a cooperative game where it is computationally hard to evaluate all coalitional values.
- To devise effective methods to compute the Shapley values for the linear production games and integer optimisation games or to approximate the values in large-scale versions of these games.

1.5 Research Contributions

We separate the main contribution of this thesis into three parts. The first contribution is the design of new cooperative games such as the generalised minimum spanning tree games and the multidimensional integer knapsack games. The second contribution is the plan and implementation of an efficient method to find a payoff allocation in the core/ least core of GMSTG. The last contribution is the theoretical approach to computing the Shapley value of LPG and MIKG.

The first contribution, the design of new cooperative games, is inspired by the observation of some regular operational research interaction in practice. The fact that many

practical networks have the clustering structure leads to the definition of the generalised minimum spanning tree games. The MIKG is the generalised version of linear production game, where the value function has the integer decision variables. We also present several theoretical results of these games and their solution concepts.

In the next contribution, we devise an original technique to compute a cost allocation for the core or least core in GMSTG. In this game, we implement the constraint generation algorithm to find a solution in the core. Our approach is different from others in the way we introduce new constraints into the relaxation problem such that the final stable allocation can be calculated without all the coalitional values.

The main contribution of this thesis is the development of novel methods to find the Shapley value for some operational research games with linear/ integer programming structures. We combine different methods of linear programming sensitivity analysis and stratified sampling technique to compute the Shapley value for large LPG. In the MIKG section, we develop the algebraic method to speed up the process of recalculating the coalition value for MIKG.

The computational results demonstrate the instances where these methods can be applied effectively in operational research games. We also interpret and illustrate the insights of the core and the Shapley value by some specific examples and practical applications.

1.6 Outline of the Dissertation

In the remainder of this thesis, we describe some subclasses of operational research games, discuss the related literature, present our approaches, and the proposed algorithms for allocating the payoff to the players. This is achieved through the course of the next chapters, which are structured as follows:

- In chapter 2, we define a new class of games, namely the generalised minimum spanning tree games and show how these games can be used to model some practical situations. We also describe an algorithm for computing a cost allocation in the core or least core of this game.
- Chapter 3 studies an important class of games namely Linear Production Games. In this section, we examine an efficient method to compute Shapley values exactly for Linear Production Games with less than 30 players. In addition, the Shapley values of large Linear Production Games with at least 30 players are approximated by new randomised algorithms.

- In chapter 4, we focus on a different class of games called multidimensional integer knapsack game. The Shapley value is utilised to allocate the payoff of the collaboration as the fair way to share the total payoff to players. We propose to use an algebraic Gröbner approach for solving the Integer Program, hence, help speed up the process of evaluating the Shapley value for the multidimensional integer knapsack game.
- The chapter 5 concludes the thesis by summarising the research contributions, showing the limitations of the findings and discusses some future research directions.
- The appendix A reintroduces some necessary background materials including notation, concepts and outcomes of cooperative game theory. In addition, the simple random sampling and stratified sampling techniques in Monte Carlo simulation will be surveyed. Finally, the appendix describes the connection of the concept of Gröbner basis in algebraic geometry and the creation of the test set for Integer Linear Program.

The interactions among three main classes of operational research games, their characteristic functions, and the particular solution concepts in each chapters are depicted in Figure 1.1.

Payoff (yield/cost) allocation
for Operational Research Games

OR Games	Chapter 3: LPG	Chapter 2: GMSTG	Chapter 4: MIKG
Characteristic function	LP	MILP	IP
Solution Concepts	Shapley value	The core/ least-core	Shapley value

Figure 1.1

Chapter 2

Generalised Minimum Spanning Tree Games

2.1 Introduction

The *minimum-cost spanning tree problem* (MSTP) is a well-known problem in network optimization, with a wide range of applications in communication, transportation, and computer networks. The standard MSTP is stated as follows: Given a weighted graph whose vertices might represent cities and whose edges serve as possible communication links with edge weights representing the cost of building a link or the length of the link, the aim is to select a set of communication links that would connect all the vertices such that the tree has the minimum total weight (Horowitz & Sahni 1978). We refer the reader to the work of Kruskal (1956) and Prim (1957) for further details and efficient algorithms to solve the problem.

The *minimum-cost spanning tree game* (MSTG) is defined on an undirected graph $G = (V, E)$ where V is the set of vertices and E is the set of edges. Each player owns a vertex and the payoff for any group of players is defined as the cost of a minimum-cost spanning tree of the subgraph corresponding to the group. Minimum-cost spanning tree games arise in cost allocation problems in which a joint enterprise can be represented as a tree that connects agents to a common source (e.g. Claus & Kleitman 1973, Bird 1976). MSTGs are cost sharing games where players need to be connected to a certain service supplier and form *coalitions*, (i.e., subsets of users/vertices) to share the cost of this service. When the cost incurred to any coalition is known, the question arises as how the total cost would be allocated among the users.

A *cost allocation problem* has the following features: there is a set $N = \{1, \ldots, n\}$ of users (e.g., residents, companies, divisions in an organization, etc.) who cooperate in the context of a joint venture (e.g., Internet cable network, emergency systems, etc.). The problem is to allocate the cost of the joint venture among all the users in a way that satisfies criteria such as fairness, stability, efficiency, etc. Recently, cooperative game theory has been used to model various cost allocation problems (see, for example, Frisk et al. 2010, Fiestras-Janeiro et al. 2011). Cooperative game theory analyses the potential grouping of players to form coalitions. It also provides mathematical tools for calculating stable cost allocations in the sense that a stable allocation prevents the collapse of the grand coalition. A brief introduction to cooperative game theory is given in the Appendix A.1, and we refer the interested readers to the books of Chalkiadakis et al. (2011) and Maschler et al. (2013) for a more detailed exposition.

Let 2^N be the set of all coalitions where $n = |N|$ is the number of players. A *cooperative game in characteristic function form* is represented by a pair $(N; c)$, where and $c: 2^N \rightarrow \mathbb{R}$ is a characteristic function with $c(M)$ being the cost incurred for a given coalition $M \subseteq N$. In the case of MSTG, let $G = (V, E)$ be a complete undirected graph where V is the vertice set including the source $\{0\}$ and E is the edge set. A player (city, customer) i is associated with a vertex $i \in N := V \setminus \{0\}$ of the graph G. For each subset (coalition) $M \subseteq N$, we define a graph $G(M) := (V(M), E(M))$ as the induced subgraph of G connecting all the vertices in the set $M \cup \{0\}$. The value $c(M)$ of a coalition M is the cost of a minimum-cost spanning tree on the subgraph $G(M)$.

Among several solution concepts of cooperative game theory, the core is a set of cost allocation where no group of players has an incentive to deviate. The core consists of all allocation vectors $x = \{x_j\}_{j \in N}$ such that (i) $\sum_{j \in N} x_j = c(N)$, and (ii) $x(M) := \sum_{j \in M} x_j \leq c(M)$ for every coalition $M \subseteq N$. Vectors in the core are natural candidates for stable cost allocations in the sense that no subset of users has an incentive to leave the grand coalition. A game may have an empty core, but even if not, generating solutions in the core may be computationally very difficult. Granot & Huberman (1981) prove that the core of a minimum-cost spanning tree game is non-empty. Furthermore, a point in the core can be calculated directly from the minimum-cost spanning tree in the problem. Later, Faigle & Kern (1997) show that, checking if a given payoff distribution is a member of the core and computing the least core for the MSTG are NP-hard problems. This was done by transforming the separation problem: "Given the vector $x \in \mathbb{R}^n$ with $x(N) = c(N)$ decide whether there exists a coalition M such that $x(M) \geq c(M)$" into the exact cover by 3-Sets problem, which is NP-complete . We will provide the formal definition of the core and the least core in Section 2.2.2.

The *generalized minimum spanning tree problem* (GMSTP) is an extension of the MSTP and it was introduced by Myung et al. (1995). Given an undirected graph whose vertices are partitioned into a number of subsets (clusters), the GMSTP is to find a minimum-cost tree which includes exactly one vertex from each cluster. Myung et al. (1995) show that the GMSTP is strongly NP-hard by using a reduction from the vertex cover problem. The authors also present four integer linear programming formulations and a branch-and-bound algorithm to solve instances of up to 100 vertices. Feremans et al. (2002) and Pop (2009) describe twelve different formulations for GMSTP and study the relationships between the polytopes of their linear relaxations.

This chapter introduces the *generalized minimum spanning tree game* (GMSTG) and proposes the computational methods for calculating its cost allocation. The GMSTG is related to the GMSTP in the same way as the MSTG is related to the MSTP. Our study of the GMSTG is motivated by the potential applications that this game finds in practice, two examples are given below:

- **Designing local area networks:** In this application, the aim is to connect a number of local networks via transmission links such as optical fibres as might be the case in metropolitan area networks (Gerla & Frata 1988) or regional area network (Prisco 1986). In this case we are seeking a minimum-cost spanning tree which selects exactly one vertex from each local network. Then, finding an optimal network design with the least cost is equivalent to solving a GMSTP while calculating the shared cost among the local areas is the GMSTG.

- **Water-supply distribution:** Consider the example shown in Fig. 2.1 in which a company supplying water in a particular region wishes to establish distribution hubs in different cities shown by A–G within the region. The hubs are connected via edges shown in bold representing the water flow, each with a fixed installation cost. There are several potential locations shown by vertices labelled 1–18 on which to place distribution hubs, but only one hub will be chosen in each city (player) depending on its power, capacity and position.

The main contributions of this chapter are: (i) to introduce GMSTG as a generalization of the MSTG and study its properties, and (ii) to describe a constraint generation algorithm to calculate stable payoffs. The rest of the chapter is organized as follows. Section 2.2 formally defines the GMSTG and discusses some properties of the game. A solution algorithm based on constraint generation is described in Section 2.3. Section 2.4 presents computational results obtained with the proposed algorithm. Conclusions are stated in Section 2.5.

2.2 Generalized Minimum Spanning Tree Game

In this section, we first provide a formal definition of the GMSTG, following which we define the core and the least core.

2.2.1 Definitions

Let $G = (V, E)$ represent an undirected graph with the vertex set V and the edge set E. Graph G is called a *graph of clusters* if the vertex set $V = \{1, \ldots, m\}$ can be partitioned into n clusters V_k with $k \in N = \{1, \ldots, n\}$ and the source vertex $\{0\} \notin V_k$ for all $k \in N$. We assume that all clusters connects to the source and there is no intra-cluster edge in the graph $G = (V, E)$.

We denote $D := \{d_{i,j}\}_{i,j \in V}$ as the cost matrix of the graph G where $d_{i,j}$ is the cost of connecting vertices i and j, and $d_{i,j} = 0$ for all $i, j \in V_k, \forall k \in N$. Suppose that the cost matrix D is included implicitly in the definition of the graph $G = (V, E)$. Moreover, if the cost of an edge is large enough, we will not sketch that edge in the represented graph (V, E) (as seen in Figure 2.1). For a subset of some clusters $S \subset N$, let the graph $(V(S), E(S))$ be the induced subgraph of G connecting the source $\{0\}$ and all vertices in clusters S.

Definition 2.1. The *Generalized Minimum Spanning Tree Problem* (GMSTP) on a graph $G = (V, E)$ of clusters is defined as the problem of finding a minimum-cost tree spanning over a subset of vertices which includes exactly one vertex from each cluster. This minimum-cost spanning tree is called the *optimal tree*, and denoted as $\tau(G)$.

The Figure 2.1 also shows an optimal tree, i.e. a solution of the GMSTP of the previous defined water-supply network.

Definition 2.2. Given a graph $G = (V, E)$ of clusters, a *Generalized Minimum Spanning Tree Game (GMSTG)* (N, c) on graph G is constructed as follows. The player set N corresponds to the set of $n = |N|$ clusters $\{V_1, \ldots V_n\}$. The characteristic function is defined by the shared cost $c(S)$ of such a coalition S is defined by the cost of a GMST of graph $G(S) = (V(S), E(S))$.

Notice that the player set in GMSTG is different to the typical cases of other cooperative games (e.g., Granot & Huberman 1981, Bergantinos & Maria 2012) where each vertex is a player. Since the players of GMSTG are different clusters of vertices, several players may have more than one vertex to choose in each coalition.

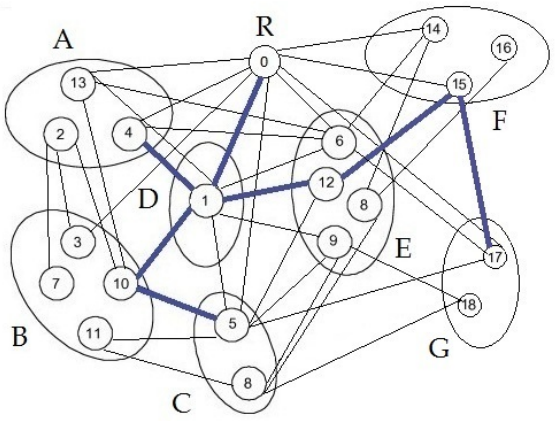

Figure 2.1: An example of a generalized water-supply network graph and a solution of the GMSTP shown by bold lines

2.2.2 The Core and Least Core of the GMSTG

In Section 2.2.3, we will show that the GMSTG has the super-additive property, i.e. a grand coalition is always formed. The question is how to share this total cost among all the players. One of the most popular solution concepts in cooperative game theory is the core, which can be given in the following form:

$$\textbf{Core}(N,c) = \{x \in \mathbb{R}^n : \sum_{i \in N} x_i = c(N); \sum_{i \in S} x_i \leq c(S),\ \forall S \subset N\},$$

where $c(S)$ is the value of the GMSTP on subgraph $(V(S), E(S))$.

In the water-supply distribution example shown in Fig. 2.1, we consider a cost allocation vector x for seven different cities. For the cost allocation x to be in the core, any group of players $S \subset N$ must have no incentive to deviate to form a coalition themselves. In other words, the cost allocated $x(S)$ should be no larger than the cost incurred $c(S)$ by forming the coalition, i.e. $x(S) \leq c(S),\ \forall\ S \subset N$. When this condition is violated, the coalition is not happy with the cost allocation and the grand coalition is not stable.

The core, however, might be empty. If this happens, the condition of no sub-coalition can gain anything by deviating can be relaxed. Assuming that a sub-coalition will deviate from the grand coalition if the gain from the change is more than the cost of deviation, one can define the ϵ-***Core*** as follows:

$$\epsilon\textbf{-Core}(N,c) = \{x \in \mathbb{R}^n : \sum_{i \in N} x_i = c(N); \sum_{i \in S} x_i \leq c(S) + \epsilon,\ \forall S \subset N\}.$$

Let $\epsilon^*(G) = \inf\{\epsilon > 0 : \epsilon\textbf{-Core}$ of the GMSTG on G-graph is non-empty$\}$. The *least core* of the GMSTG on G-graph is its $\epsilon^*(G)$-Core and $\epsilon^*(G)$ is called the value of the least core of the GMSTG on graph G. The least core can be found by solving the following linear programming problem

$$\begin{aligned} & \min_{\epsilon,x} && \epsilon \\ \text{s.t.} \quad & \epsilon && \geq x(S) - c(S),\ \forall S \subset N, \\ & x(N) && = c(N). \end{aligned} \tag{2.1}$$

One of the nice properties of the least core is non-emptiness and if the value of the least core problem (2.1) is zero then the core of the game exists and coincides with the least core.

For the MSTG, some heuristic methods have been proposed to calculate a cost allocation in the core such as the Bird rule (Bird 1976), the obligation rule (Tijs et al. 2006) and Folk solution (Bogomolnaia & Moulin 2010). The *Bird rule* of a MSTG on the graph G is described as follows: initially, one has to find a minimum-cost spanning tree $\rho(G)$ by using Kruskal's or Prim's algorithms. Then the Bird rule's allocation corresponding to $\rho(G)$ assigns for each player (vertex) the edge cost which connects that player with her immediate predecessor in $\rho(G)$.

In the case of the GMSTG, the optimal tree $\tau(G)$ is used instead of the minimum-cost spanning of MSTG in the definition of Bird rule. We will show in the example below that the Bird rule's allocation for the GMSTG may not stay in the core.

Example 2.1. *Consider a water supply network game with the root vertex* $R = \{0\}$ *and 3 players (cities)* $N = \{A, B, C\}$*, where the connection costs are shown in Fig. 2.2. Each player has one option for choosing the distribution hub, with the exception of player A who can choose between vertices* $\{1\}$ *or* $\{2\}$*. We assume that edges not shown in the graph do not exits, i.e., with infinite cost. The characteristic function for this game shown in Table 2.1.*

Coalition S	$\{\emptyset\}$	$\{A\}$	$\{B\}$	$\{C\}$	$\{A,B\}$	$\{A,C\}$	$\{B,C\}$	$\{A,B,C\}$
$c(S)$	0	10	50	35	45	30	70	70

Table 2.1: The characteristic function of water-supply network games with three players

The following set of inequalities describe the individual rationalities and subgroup rationalities of the GMSTG core:

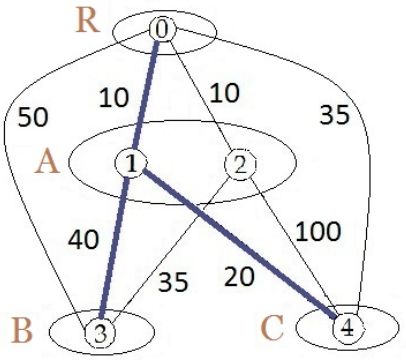

Figure 2.2: A water distribution network with a solution of the GMSTP shown by bold lines

$$\begin{aligned} &x_A \leq 10, \quad x_B \leq 50, \quad x_C \leq 35, \\ &x_A + x_B \leq 45, \quad x_A + x_C \leq 30, \quad x_B + x_C \leq 70, \\ &x_A, x_B, x_C \geq 0. \end{aligned} \tag{2.2}$$

Hence the core of this GMSTG is defined by

$$\textbf{\textit{Core}}(N,c) := \{x \in \mathbb{R}^3 | x_A + x_B + x_C = 70 \text{ and inequalities (2.2) hold.}\}$$

We can quickly find a cost allocation such as $x = (5, 40, 25)$ *which lies in the core. However, the solution* $x = (10, 40, 20)$ *yielded by the Bird rule is not in the core of the GMSTG, as it violates the constraint* $x_A + x_B \leq 45$ *in the inequality set (2.2).*

One other important feature of the GMSTG is that, unlike the MSTG, its core might be empty. As an example, if we modify the game above such that the cost $d_{2,3}$ *of the edge* $\{2, 3\}$ *that connects player A and player B is changed from* 35 *to* 25*, the core of the game becomes empty. This can be seen from the constraint set* $x_A + x_B \leq 35$ *,* $x_A + x_C \leq 30$ *and* $x_A + x_B + x_C = 70$*, which does not have a non-negative solution vector.*

Hence, the Bird rule for finding a good cost allocation in the MSTG does not apply to the GMSTG problem. Since there exists GMSTG with empty core in Example 2.1 where $d_{2,3} = 25$, other solution concepts such as the obligation rule and Folk solution do not generally apply.

2.2.3 Properties of the GMSTG and its core

In this section, we present a number of properties of the GMSTG and its core. The first one concerns the super-additivity.

Lemma 2.3. *GMSTG is a super-additive game, i.e.,* $c(S_1 \cup S_2) \leq c(S_1) + c(S_2)$ *for* $S_1, S_2 \subseteq N$ *and* $S_1 \cap S_2 = \{\emptyset\}$.

Proof. Let S_1 and S_2 be two disjoint subsets of clusters and let $c(S_1), c(S_2), c(S_1 \cup S_2)$ be the values of the solutions of the GMSTP defined on graphs (V_{S_1}, E_{S_1}), (V_{S_2}, E_{S_2}) and $(V_{S_1 \cup S_2}, E_{S_1 \cup S_2})$ respectively. As we can always generate a spanning tree on $(V_{S_1 \cup S_2}, E_{S_1 \cup S_2})$ by combining the two spanning trees on two subgraphs (V_{S_1}, E_{S_1}) and (V_{S_2}, E_{S_2}) with a value equal to $c(S_1) + c(S_2)$, we have $c(S_1 \cup S_2) \leq c(S_1) + c(S_2)$ by the definition of minimum-cost spanning tree. □

Remark 2.4. Because the sum of the individual costs of the coalitions is no less than the cost of a union of disjoint coalitions, the grand coalition will be formed for the benefit of all players.

In the next part, we will define the induced subgraph of G on a vertex set T, its subgame, and the optimal tree game. Afterwards, the relationship between its core and the core of GMSTG is described.

Let $T \subset V$ be a subset of vertices including the source $\{0\}$ in the graph $G = (V, E)$. A coalition $M := cls(T) \subset N$ is defined as the minimal set of clusters (players) containing all the vertices in T. Therefore, the set T contains at least one vertex Q_k from each cluster V_k, $\forall k \in M$. The graph $G_T := (T, E_T)$ is constructed as the *induced subgraph* of G to the set T.

For each coalition of players $S \subset M$, let $T(S)$ be the set of vertices belonging to the coalition S in the subgraph G_T. The value $c^T(S)$ is set as the cost of the MSTP on the subgraph $G_T(S) := (T(S), E_T(S))$.

Definition 2.5. Given the graph $G = (V, E)$, the *GMST-subgame* (T, c^T) is defined as the GMSTG on vertex set $\{T\}$ with the player set $M = cls(T)$ and cost function c^T.

In this definition, when the context is clear, the T notation is used as the set of players instead of $cls(T)$. In a particular case, if the set T is equal to the set of vertices in a coalition M, i.e. $T = V(M)$, we will denote the GMSTG on the coalition M as GMST-subgame (M, c^M) on the subgraph $G(M) := (V(M), E(M))$.

Example 2.2. *Given the graph $G = (V, E)$ in Figure 2.1, if we consider $T \subset V$ as the subset of vertices $\{2, 3, 5, 6, 8, 9, 10, 11, 12, 13\}$ and the source vertex $\{0\}$. The induced subgraph $G_T := (T, E_T)$ is shown in Figure 2.3.*

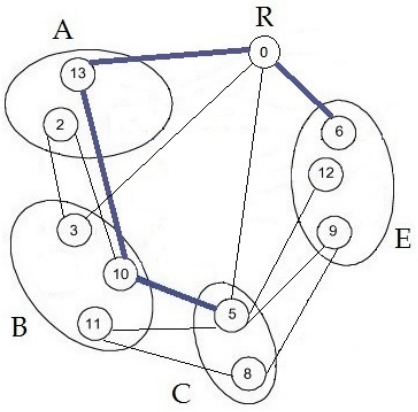

Figure 2.3: An induced subgraph example of the graph in Fig. 2.1 on a subset T and the optimal tree shown by the bold lines

Since the clusters set of T reduced to $M = \{A, B, C, E\}$, the GMST-subgame (T, c^T) has four players. For a coalition of players such as $S = \{A, B\}$, the value of $c^T(S)$ is $d_{0,13} + d_{13,10}$. When S is the grand coalition M of this subgame, $c^T(M) = d_{0,13} + d_{13,10} + d_{10,5} + d_{0,6}$ is the optimal tree cost.

Definition 2.6. Let $T^* := \{Q_1^*, Q_2^*, \ldots, Q_n^*\}$ be the set of vertices appearing in an optimal tree solution of the GMSTP on $G = (V, E)$. The GMST-subgame (T^*, c^{T^*}) is called as the *optimal tree game* (OTG) (T^*, c^{T^*}) of graph G.

Since the minimum-cost spanning tree generated by the 'Bird rule' on the optimal tree game will be the same as the optimal tree $\tau(G)$ of the GMSTG, let us call $\{\boldsymbol{x}_B\}$ the cost allocation corresponding to $\tau(G)$. If we reduce the graph G to contain only vertices in T^*, the GMSTG becomes the MSTG and the 'Bird rule' could be adapted to find a stable cost allocation.

Example 2.3. *From the simple water network Example 2.1, the optimal tree is shown in the bold line below. Therefore, the optimal tree games is a classical type of Minimum-cost Spanning Tree Game, where the Bird rule of this MSTG is $x_B = \{10, 40, 20\}$. We can check that it is exactly the Bird rule's allocation of the GMSTG.*

In what follows, we show a relationship between the cores of the OTG and the GMSTG.

Proposition 2.7. *The core of GMSTG (N, c) is a subset of the core of the optimal tree game (T^*, c^{T^*}).*

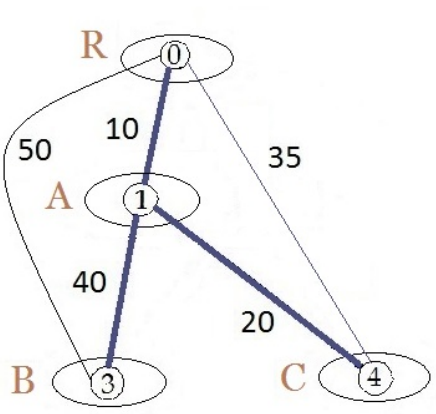

Figure 2.4: The optimal tree games constructed from the network in Figure 2.2

Proof. Assume the GMSTG-core is non-empty, i.e., there is a cost allocation $\boldsymbol{x}^* = \{x_1, \ldots, x_n\}$ that satisfies all constraints in the core problem below:

$$\begin{aligned} &x(S) \leq c(S), \qquad \forall S \subset N, \\ &x(N) = c(N). \end{aligned} \tag{2.3}$$

Consider the optimal tree game on $G = (V, E)$ generated by $\{0\}$ and $\{T^*\}$, we will prove that the vector $\boldsymbol{x}$ also satisfies all the constraints in core problem of the optimal tree game (T^*, c^{T^*}). For an arbitrary group of players $S \subset N$, $c^{T^*}(S)$ is the cost that a coalition S has to pay in the optimal tree game.

By definition, $c(S)$ is the minimum-cost spanning tree in the graph $G_S = (S, E_S)$ for any arbitrary $S \subset N$ leads to the following: $c(S) = \min_{T \in \Xi} c^T(S) \leq c^{T^*}(S)$. where $\Xi = \{T : T = \{Q_k\}_{k=1}^n,\ Q_k \in V_k, \forall k \in \{1, \ldots, n\}\}$. Therefore, $x(S) \leq c(S) \leq c^{T^*}(S)$ for all coalition S of the optimal tree game and therefore the cost allocations $\boldsymbol{x}$ belongs to the core of the optimal tree game (T^*, c^{T^*}). □

Remark 2.8. The demonstration of the proper subset property for these two core sets is shown in Figure 2.2, where the Bird allocation $\boldsymbol{x}_B = \{10, 40, 20\}$ is in the core of the OTG (T^*, c^{T^*}), but not in the core of the GMSTG.

When defining the least core, the stability constraints in the definition of the core are relaxed. An alternative approach is to relax the feasibility constraint $\sum_{i=1}^{n} x_i = c(N)$. This happens when an external party wishes to stabilize the game, by offering a subsidy amount Δ to the grand coalition if all players collaborate as a large group. More formally, given a super-additive game (N, c) and $\Delta \geq 0$, let $G^\Delta = (N, c^\Delta)$ be cooperative game over the set of players N with the characteristic function given by $c^\Delta(N) = c(N) - \Delta$

and $c^{\Delta}(S) = c(S)$ for all $S \subset N$. Then, the core set of cost allocation for the game G^{Δ} is defined as follows:

$$\begin{aligned} \mathbf{Core}(N, c^{\Delta}): \quad & x(S) \quad \leq c(S), \ \forall S \subset N, \\ & x(N) \quad = c^{\Delta}(N). \end{aligned} \tag{2.4}$$

In general, the GMSTG is not monotone, i.e. the cost function c is not a monotone function. Fig. 2.5 shows a subgraph of previous water-supply network defined on a coalition M including clusters $\{A, B, C, E\}$. We notice that the solution of the induced GMSTP is not a subtree of the optimal tree $\tau(G)$. Moreover, $c(M) \geq c(N)$ might happen for such coalition $M \subset N$.

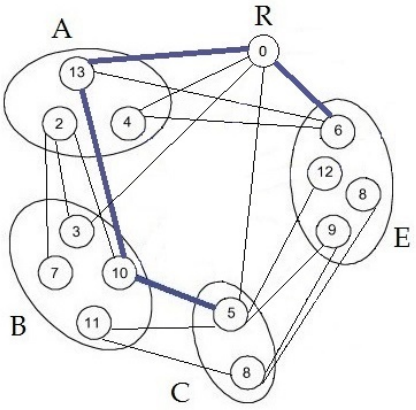

Figure 2.5: A subnetwork example of coalition M reduced from the original graph shown in Fig. 2.1 and an optimal tree shown by the bold lines

For each coalition $M \subset N$, let $\Gamma_M := c(M)$ and $\Lambda_M := \max\{c(M) - c(N), 0\}$. We now consider an amount Δ_M such that $\Lambda_M \leq \Delta_M \leq \Gamma_M$. From the GMST-subgame (M, c^M), we define a corresponding game (M, c^{Δ_M}) in a similar manner as the previous part. The characteristic function of this game satisfies $c^{\Delta_M}(M) = c(M) - \Delta_M \leq c(M) - \Lambda_M \leq c(N)$ and $c^{\Delta_M}(S) = c(S)$, $\forall S \subset M$. We can define the core set of the game (M, c^{Δ_M}) of players in coalition M by the following model:

$$\begin{aligned} \mathbf{Core}(M, c^{\Delta_M}): \quad & x(S) \quad \leq c^{\Delta_M}(S), \ \forall S \subset M, \\ & x(M) \quad = c^{\Delta_M}(M). \end{aligned} \tag{2.5}$$

After calculating the core of this subgame (M, c^{Δ_M}), for each of its cost allocation $\boldsymbol{x}_M^{\Delta_M}$ we could build a class $\{\boldsymbol{x}\}_G^{\Delta_M}$ of correspondent payoff distributions for the GMSTG (N, c) by using the amount $c(N) - c(M) + \Delta_M \geq 0$ divided arbitrarily to players in $N \setminus M$.

This amount is feasible due to the condition $\Delta_M \geq \Lambda_M := \max\{c(M) - c(N), 0\}$ and $\Delta_M \leq c(M)$, and the core of GMSTG (N, c) is exactly $\{\boldsymbol{x}\}_G^{\Delta_N}$ when $\Delta_N = 0$. The process generates a set of GMSTG cost allocations denoted by $\textbf{C-set}(G, M)$. More formally,

$$\textbf{C-set}(G, M) := \bigcup_{\forall \Delta_M, \Gamma_M \geq \Delta_M \geq \Lambda_M} \{\boldsymbol{x}\}_G^{\Delta_M}.$$

Notice that the $\textbf{C-set}(G, M)$ does not depend not Δ_M since Λ_M and Γ_M are fixed for each coalition M.

Proposition 2.9. *For coalitions $S_2 \subset S_1 \subset N$, we have the relationship of **C-sets** as follows:*

$$\textbf{\textit{C-set}}(G, S_1) \subset \textbf{\textit{C-set}}(G, S_2).$$

Proof. From the **C-set** definition, we have $\textbf{C-set}(G, S_1) = \bigcup_{\Delta_{S_1} \geq \Lambda_{S_1}} \{\boldsymbol{x}\}_G^{\Delta_{S_1}}$. For each $\Delta_{S_1} \in [\Lambda_{S_1}, \Gamma_{S_1}]$, there will exist Δ_{S_2} such that $\Gamma_{S_2} \geq \Delta_{S_2} \geq \Lambda_{S_2}$, and $\{\boldsymbol{x}\}_G^{\Delta_{S_1}} \subseteq \{\boldsymbol{x}\}_G^{\Delta_{S_2}}$.

To prove that, let $\boldsymbol{x}^*$ be the solution in $\{\boldsymbol{x}\}_G^{\Delta_{S_1}}$, i.e., the subvector $\boldsymbol{x}^*_{S_1} = \{x_i^*\}_{i \in S_1}$ is in core of the game $(S_1, c^{\Delta_{S_1}})$. Because $S_2 \subset S_1$, we can define $\Delta_{S_2} := c(S_2) - x^*(S_2) = c(S_2) - \sum_{i \in S_2} x_i^*$. For any coalition S that satisfies $S \subset S_2 \subset S_1$, we have $x^*(S) \leq c(S) = c^{\Delta_{S_2}}(S)$ and $x^*(S_2) = c(S_2) - \Delta_{S_2} = c^{\Delta_{S_2}}(S_2)$. Therefore, subvector $\{x^*_{S_2}\}$ is also in the core of the game $(S_2, c^{\Delta_{S_2}})$, i.e, x^* is the solution in $\{\boldsymbol{x}\}_G^{\Delta_{S_2}}$. □

The results of Proposition 2.7 and Proposition 2.9 are combined to show the relationship among the the GMSTG core, the OTG core and the C-set in Figure 2.6.

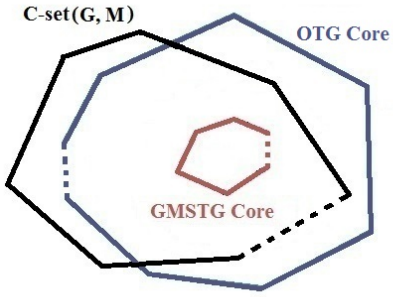

Figure 2.6: The relationship among the GMSTG core, the OTG core and the C-set

We now present another proposition showing the relationships between the GMSTG core and its related **C-sets** as follows:

Proposition 2.10. *The core of GMSTG* (N, c) *is the intersection of the* ***C-set***(G, M) *where* $M = N \setminus \{i\}$*, for all clusters (players)* $\{i\} \in N$*, i.e.,*

$$\boldsymbol{Core}(N, c) = \bigcap_{i=1}^{n} \boldsymbol{C\text{-}set}(G, N \setminus \{i\}).$$

Proof. If there exists a cost allocation $\{\boldsymbol{x}\}$ stays in $\mathbf{C\text{-}set}(G, N \setminus \{i\})$ for $\forall i \in N$, then $\{\boldsymbol{x}\} \in \mathbf{C\text{-}set}(G, M)$ for any coalition $M \subset N$ because of Proposition 2.9. This payoff distribution will satisfy constraints $x(M) = c^{\Delta_M}(M) \leq c(M)$ for any subsets $M \subset N$. Therefore, $x(S) \leq c(S)$ for all $S \subset N$ and $x(N) = c(N)$, i.e, $\boldsymbol{x} \in \mathbf{Core}(N, c)$.

We now assume $\{\boldsymbol{x}'\}$ is a cost allocation belonging to the core of GMSTG (N, c). For any subset $M \subset N$, we will prove that $\boldsymbol{x}' \in \mathbf{C\text{-}set}(G, M)$. Because $x'(M) \leq x'(N) = c(N)$ and $x'(M) \leq c(M)$, we set $\Delta'_M := c(M) - x'(M) \geq \max\{c(M) - c(N), 0\}$. For any $S \subset M \subset N$, $x'(S) \leq c(S) = c^{\Delta'_M}(S)$ and $x'(M) = c(M) - \Delta'_M = c^{\Delta'_M}(M)$. Therefore $\boldsymbol{x}' \in \{\boldsymbol{x}\}_G^{\Delta'_M} \subset \mathbf{C\text{-}set}(G, M)$.

□

Remark 2.11. If there exists one coalition $M \subset N$ of players such that the $\mathbf{C\text{-}set}(G, M)$ is empty, then the core of GMSTG (N, c) is also empty.

By solving the GMSTP, each general graph $G = (V, E)$ can always generate an optimal tree $\tau(G)$. We now consider the class $\Omega(N)$ of graphs G such that each graph has n clusters and the optimal tree $\tau(G)$ has $p \geq 2$ edges incident to the source vertex 0 (denote as the p branches property of the optimal tree). Afterwards, p GMST-subgames (N^i, c^{N^i}) are constructed for $i = \{1, \ldots, p\}$. Note that as $c^{N^i}(S) = c(S)$ for all $S \subseteq N^i$, the notation c is adopted instead of c^{N^i}. We will prove that the core of the GMSTG (N, c) is a proper subset of the Cartesian product (Π) of the cores of these p GMST-subgames.

Remark 2.12. The condition that a graph $G = (V, E)$ with the optimal tree $\tau(G)$ has more than one edges incident to the source vertex $\{0\}$ is equivalent to the existence of a partition of N clusters into subsets of clusters $\{N^1, \ldots, N^p\}$ satisfying $c(N) = \sum_{i=1}^{p} c(N^i)$). We now call this condition of the partition of N as *efficient coalition structure.*

An example of such a graph partition is shown in Figure 2.7.

For each branch $i \in \{1, \ldots, p\}$ of the optimal tree $\tau(G)$, let denote (N^i, c) as the GMST-subgame defined by the set of clusters $\{0\} \cup N^i$ and cost function $c(S, N^i) = c(S)$, $\forall S \subset N^i$. The core of the GMST-subgame (N^i, c) is presented in following models:

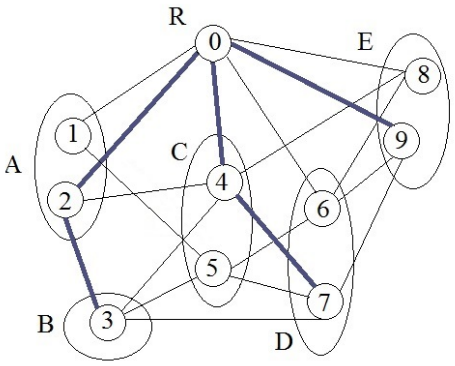

Figure 2.7: A simple graph in $\Omega(N)$ with the optimal tree $\tau(G)$ having $p = 3$ branches and an optimal tree shown by bold lines

$$\begin{aligned} \textbf{Core}(N^i, c): \quad x^i(S) &\leq c(S),\ \forall S \subset N^i, \\ x^i(N^i) &= c(N^i), \end{aligned} \tag{2.6}$$

where n_i is the number of clusters in N^i and $\boldsymbol{x}^i = \{x^i_j\}_{j=1}^{n_i}$.

Definition 2.13. The Cartesian product (Π) of the cores of these p GMST-subgames is defined as follows:

$$\begin{aligned} \Pi: \quad \textbf{Core}(N^1, c); \ldots; \textbf{Core}(N^p, c) &\rightarrow \prod_{i=1}^{p} \textbf{Core}(N^i, c) \\ \Pi(\boldsymbol{x}^1, \ldots, \boldsymbol{x}^p) &= \boldsymbol{x}^* \end{aligned} \tag{2.7}$$

such that $\boldsymbol{x}^*_j = \boldsymbol{x}^i_j$ for all player j in coalition N^i, $\forall i = \{1, \ldots, p\}$.

Theorem 2.14. *If $\{N^1, \ldots, N^p\}$ is an efficient coalition structure of the GMSTG (N, c) with $p \geq 2$, the core of the GMSTG (N, c) is a proper subset of the Cartesian product of the cores of the GMST-subgame (N^i, c) for $i = 1, \ldots, p$, i.e.,*

$$\boldsymbol{Core}(N, c) \subset \prod_{i=1}^{p} \boldsymbol{Core}(N^i, c) \subset \boldsymbol{C\text{-}set}(G, N^i).$$

Proof. Let $\boldsymbol{x}^* = \{x^*_1, \ldots, x^*_n\}$ be a payoff vector in the core of GMSTG (N, c). We will prove that $\boldsymbol{x}^* = \prod_{i=1}^{p} x^{*i}$, where $\boldsymbol{x}^{*i} := \{x^*_j\}_{j \in N^i}$ is a cost payoff of the GMST-subgame (N^i, c). Because $\{N^1, \ldots, N^p\}$ is an efficient coalition structure of the GMSTG (N, c), then $c(N) = \sum_{i=1}^{p} c(N^i)$. Moreover, the definition of the core has the condition $x^*(N^i) \leq c(N^i)$ for all $i = 1, \ldots, p$ and $x^*(N) = c(N)$. Hence, the inequality in sequence $x^*(N) = \sum_{i=1}^{p} x^*(N^i) \leq \sum_{i=1}^{p} c(N^i) = c(N)$ becomes equality, and $x^*(N^i) = c(N^i)$ for all

$i = 1, \ldots, p$. In particular, $\boldsymbol{x}^{*i}$ satisfies all constraints in the core formulation (2.6). Therefore, $\boldsymbol{x}^*$ belongs to the Cartesian product of the cores of the GMST-subgame (N^i, c) for $i = 1, \ldots, p$. To show that the inclusion is strict, we provide an example in Section 2.4.1, where $\mathbf{Core}(N, c) \neq \prod_{i=1}^{p} \mathbf{Core}(N^i, c)$.

Let $\boldsymbol{y}$ is a payoff vector in $\prod_{i=1}^{p} \mathbf{Core}(N^i, c)$. For any $i \in \{1, \ldots, p\}$, we have $y(S) \leq c(S)$, $\forall S \subset N^i$ and $y(N^i) = c(N^i)$. Because of the efficient coalition structure property, $\sum_{i=1}^{p} y(N^i) = \sum_{i=1}^{p} c(N^i) = c(N)$. Combined with the definition of **C-set**, hence, $\boldsymbol{y}$ is also a payoff vector in $\mathbf{C\text{-}set}(G, N^i)$ for any $i = \{1, \ldots, p\}$.

□

Remark 2.15. If a graph G has an optimal tree $\tau(G)$ such that there exists $i \in \{1, \ldots, p\}$ where $\mathbf{Core}(N^i, c) = \emptyset$, then the core of the GMSTG (N, c) is also empty. This property provides a method to check the emptiness of the core for some large-sized GMSTGs.

2.3 Computational Methods for Finding the Core and the Least Core

The problem of finding a feasible cost allocation in the least core is difficult because its mathematical programming formulation has 2^n constraints, which would necessitate the solution of 2^n GMSTPs just for the purpose of obtaining the input of the resulting optimization problem. Therefore, brute-force techniques that attempt to solve 2^n GMSTPs and then solve an LP with 2^n constraints would be impractical to solve GMSTG instances when the number of clusters is $n \geq 8$. To overcome this difficulty, we now present a constraint generation algorithm to solve the GMSTG, and start by describing a formulation of the GMSTP that is used within the algorithm.

2.3.1 A multi-commodity GMSTP Formulation

Among the four existing GMSTP formulations presented by Myung et al. (1995), four formulations by Feremans et al. (2002) and four others in Pop (2009), we consider a multi-commodity flow formulation defined on a directed graph $D = (V, A)$ of the first paper. The motivation behind the choice of this particular model is its compact form (with a polynomial number of constraints) compared to other models (with an exponential number of constraints). Moreover, the formulation yields the best linear

relaxation among others because of its polytope structure. Therefore it arises as a natural candidate for the construction of the exact algorithm for the separation problem in Section 2.3.2.

This model describes a network flow of n different commodities from the source $\{0\}$ to the sinks in n clusters. For the commodities $k \in N$, it has a total one unit supply at a source $\{0\}$ and an unit demands at one sink vertex in the cluster k. For each vertex $i \in V_k$, let y_i be the amount of commodity k arriving at vertex i, hence, variables y_i are defined on all vertices of $V \setminus \{0\}$. Therefore, the total demands $\sum_{i \in V_k} y_i$ of commodity k in the cluster V_k is one unit. A binary variable w_{ij} is defined which equals 1 if and only if vertices i and j are connected on the optimal tree, and 0 otherwise. Continuous variables f_{ij}^k are used as the amount of commodity k flow on arc (i, j) travelling from the source $\{0\}$ to the cluster V_k, for all $k \in N$.

This multi-commodity network flow formulation has a polynomial number of constraints and is shown in the following:

$$\mathbf{P_1}: \quad \min_{w_{ij}, y_i, f_{ij}^k} \quad \sum_{(i,j) \in A} c_{ij} w_{ij} \tag{2.8a}$$

$$\text{s.t.} \quad \sum_{i \in V_k} y_i = 1, \ \forall k \in N \cup \{0\}, \tag{2.8b}$$

$$\sum_{i \in V_k} \sum_{j=1}^{m} w_{ji} \leq 1, \ \forall k \in N, \tag{2.8c}$$

$$\sum_{j=1}^{m} f_{ij}^k - \sum_{j=1}^{m} f_{ji}^k = \left\{ \begin{array}{ll} 1, & i = \{0\}, \\ -y_i, & \forall i \in V_k, \\ 0, & \text{otherwise} \end{array} \right\}, \forall k \in N, \tag{2.8d}$$

$$0 \leq f_{ij}^k \leq w_{ij}, \ \forall (i,j) \in A, \forall k \in N, \tag{2.8e}$$

$$y_i, w_{ij} \in \{0, 1\}, \ \forall i, j \in V. \tag{2.8f}$$

The first set of constraints (2.8b) guarantees that only one vertex is chosen as the sink from each cluster V_k. The set of constraints in (2.8c) allows for at most one incoming arc to each cluster. The set of constraints (2.8d) ensures flow conservation at all intermediate vertices $\{i\} \in V$. The final set of constraints in (2.8e) models the relationship between the design and the flow variables.

2.3.2 Constraint Generation Algorithm

We now describe a constraint generation algorithm to solve the (least) core problem. The motivation for using this technique is to resolve two technical issues: (a) the LP formulation for solving the least core has an exponentially large number of constraints and (b) the input of this LP requires solving 2^n GMSTPs of $c(S)$, $\forall S \subset N$. The main idea behind this algorithm is to solve problem (2.1) only with a limited subset of constraints $E_0 \subset 2^N$ at the beginning rather than solving the whole problem with all 2^n constraints included and later add only some necessary constraints $x(S) \leq c(S) + \epsilon$ into the system. At each step t, the algorithm initially solves the following restricted problem:

$$\begin{aligned} \mathbf{LP_t}: \quad & \min_{\epsilon^t, \boldsymbol{x}} \quad \epsilon^t \\ \text{s.t.} \quad & \epsilon^t \geq x(S) - c(S), \ \forall S \in E_t \\ & x(N) = c(N). \end{aligned} \tag{2.9}$$

After solving problem (2.9), we obtain a new solution $x^t = (x_1^t, \ldots, x_n^t)$ and the value ϵ^t of $\mathbf{LP_t}$. In the separation step, we identify the most violated constraint in least core problem (2.1) by solving the following optimization problem with a fixed vector x^t:

$$\mathbf{P_2}: \quad \min_{S \subseteq N; S \neq \emptyset} \quad c(S) - x^t(S). \tag{2.10}$$

A coalition S is described by the binary n-vector $s = \{s_k\}_{k \in N}$, where $s_k \in \{0, 1\}$ depends on whether the player $\{k\}$ is a member of the coalition S. Applying the multi-commodity flow formulation $\mathbf{P_1}$ of $c(S)$ to problem (2.10), the separation problem can be written in the form of a MILP with binary variables s_k, continuous variables f_{ij}^k and binary variables w_{ij}, y_i as shown in the following:

$$
\begin{aligned}
\mathbf{P_3}: \quad & \min_{f_{ij}^k, w_{ij}, y_i, s_k} \quad \sum_{(i,j)\in A} c_{ij} w_{ij} - \sum_{k\in N} s_k x_k^t \\
\text{s.t.} \quad & \sum_{i\in V_k} y_i = s_k, \ \forall k \in N \cup \{0\}, \\
& \sum_{i\in V_k} \sum_{j=1}^{m} w_{ji} \leq s_k, \ \forall k \in N, \\
& \sum_{j=1}^{m} f_{ij}^k - \sum_{j=1}^{m} f_{ji}^k = \left\{ \begin{array}{ll} s_k, & i = \{0\}, \\ -y_i, & \forall i \in V_k, \\ 0, & \text{otherwise} \end{array} \right\}, \forall k \in N, \\
& 0 \leq f_{ij}^k \leq w_{ij}, \ \forall (i,j) \in A, \forall k \in N, \\
& y_i, w_{ij} \in \{0,1\}, \ \forall i,j \in V, \\
& s_k \in \{0,1\}, \ \forall k \in N.
\end{aligned}
\tag{2.11}
$$

Compared to $\mathbf{P_1}$, formulation $\mathbf{P_3}$ has a new set of variables s which changes both the objective function and most of the constraints. Hence, the new formulation becomes more difficult to solve.

After the separation step, if the optimality of the problem (2.1) is not reached, the solution s of $\mathbf{P_3}$ is used to generate a new constraint. In particular, suppose $\mathbf{P_3}$ has an optimal solution s^t, then the coalition $S^t = \{s_j^t\}$ is the one that receives the least negative excess. A new constraint in the form $\epsilon \geq x(S^t) - c(S^t)$ is then added to the relaxed problem (2.9), which is resolved to produce a new set x^{t+1} of variables, and the algorithm continues in this manner.

A pseudo-code of the constraint generation algorithm for calculating the GMSTG least core is given in Algorithm 1.

Algorithm 1 Constraint Generation Algorithm

Initialize: $maxiter = 10^4$; $\omega = 10^{-6}$; iteration $t = 1$;
Constraint set $E_1 = E_0 \subset F$;
repeat
Compute solution x^t and value ϵ^t of $\mathbf{LP_t}$;
Find S^t in F such that:

$$S^t = \underset{S}{\operatorname{argmin}}\{c(S) - x^t(S)\};$$

$\overline{\epsilon}(t) = x^t(S^t) - c(S^t)$;
$\underline{\epsilon}(t) = \epsilon^t$;
if $|\overline{\epsilon}(t) - \underline{\epsilon}(t)| \leq \omega$ **then**
stop;
end if
$E_t = E_t \cup \{S^t\}$;
$t := t + 1$;
until $t = maxiter$.

To solve problem (2.10), we use CPLEX 12.5.0 with Matlab interface to obtain the optimal set S^t. This approach is useful if the process of finding the coalition S^t that minimizes $(c(S) - x^t(S))$ over F is fast enough. In fact, this is one of the reason why the multi-commodity flow model $\mathbf{P_1}$ is used as a basis for formulating the separation problem.

We also have some inequalities in step t as following:

$$\underline{\epsilon}(t) := \epsilon^t \leq \text{val}(LP(2.1)) \leq \overline{\epsilon}(t) := x^t(S^t) - c(S^t),$$

where val$(LP(2.1))$ is the optimal value of the corresponding least core linear program. In the iteration t, the lower bound ϵ^t is the objective value of the restricted problem (2.9), hence, it is an increasing lower bound.

For a small enough ω, this procedure stop when $|x^t(S^t) - c(S^t) - \epsilon^t| \leq \omega$. This happens because adding any new constraints to problem (2.9) will not change the objective value ϵ^t. Since its total number of constraints is 2^n, the constraint generation algorithm will always terminate. In practice, the number of constraints generated in the algorithm will be substantially smaller, as we show in the numerical result (Section 2.4.2).

2.3.3 Other approaches for cost sharing

In cooperative game theory, there are a few alternative mathematical formulations for calculating stable solutions. Sometimes, an optimal solution of the least core problem is such that some players are disadvantaged. This may happen, for example, when there are large differences between the shares of pairs of players. One way to avoid such a situation is called an equal profit method (Frisk et al. 2010), which is used to minimise the differences between the ratios of cost share over individual cost for every pair of players. Therefore we consider a model to find a stable cost allocation, such that the maximum difference in pairwise relative costs is minimised. This model will take as input an optimal solution ϵ^* of the least core problem and is presented below:

$$
\begin{aligned}
\hat{\mathbf{P}}(\epsilon^*): \quad & \min_{x,f} \quad f & & && \text{(2.12a)}\\
\text{s.t.} \quad & f & \geq \frac{x_i}{c(\{i\})} - \frac{x_j}{c(\{j\})}, \quad \forall i,j \in N, && \text{(2.12b)}\\
& \sum_{i\in S} x_i & \leq c(S) + \epsilon^*, \quad \forall S \subset N, && \text{(2.12c)}\\
& \sum_{i\in N} x_i & = c(N), && \text{(2.12d)}\\
& x_i & \geq 0, \quad \forall i \in N. && \text{(2.12e)}
\end{aligned}
$$

The new model $\hat{\mathbf{P}}(\epsilon^*)$ is an LP, and could be solved by adapting the CGA described earlier for a fixed ϵ^*. Because of the definition of the least core ϵ^*, there always exits a cost allocation x^* which satisfies constraints (2.12c)– (2.12e). Therefore, the feasible region of problem (2.12) is non-empty. The difference of the new CGA algorithm is in the restricted model, where we add an extra n^2 constraints (2.12b) into the LP formulation at each step t of $\widehat{\mathbf{LP_t}}(\epsilon^*)$, given below:

$$
\begin{aligned}
\widehat{\mathbf{LP_t}}(\epsilon^*): \quad & \min_{x,f_t} \quad f_t & &\\
\text{s.t.} \quad & f_t & \geq \frac{x_i}{c(\{i\})} - \frac{x_j}{c(\{j\})}, \quad \forall i,j \in N,\\
& \sum_{i\in S} x_i & \leq c(S) + \epsilon^*, \ \forall S \in E_t\\
& \sum_{i\in N} x_i & = c(N),\\
& x_i & \geq 0, \quad \forall i \in N.
\end{aligned}
\qquad \text{(2.13)}
$$

The outcome of the algorithm will generate new payoff vector $\boldsymbol{x}^{*\prime}$.

In the next section, we provide some computational results on the algorithm and formulation described above and present some comparative results. An illustrative example of the equal profit model will also be shown in Section 2.4.1.

2.4 Numerical Results

This section first presents results obtained with the constraint generation algorithm on a small-scale problem instance, and extends the results to a set of larger size instances.

Our CGA algorithm was implemented in MATLAB and run on an Intel-core i5 PC with 2.6 GHz CPU and 4GB RAM.

2.4.1 An illustrative example

Example 2.4. ***(Internet network)***

This example is motivated by an application in telecommunication networks, and concerns an internet provider located at vertex 0 *wishing to establish its hubs (or gateways) at four different cities displayed as* K, L, M *and* N*, by setting up one hub in each city. Each city* k *will have some possible locations to set up its hub as shown in the Fig 2.8. The internet provider can connect the potential hubs using optical fibres, where the costs between every pair of vertices are as shown in Table 2.2. For edges that do not exist in the graph, the corresponding cost in this table is shown as "—".*

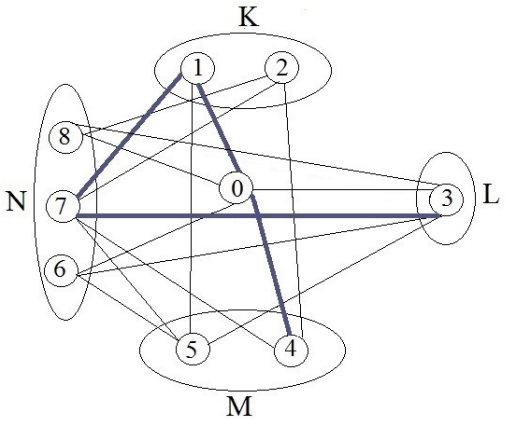

Figure 2.8: An internet-cable network graph and the solution of GMSTP shown by bold lines

Vertices	0	1	2	3	4	5	6	7	8
0	0	89	—	514	114	—	385	—	315
1	89	0	0	—	—	296	—	40	—
2	—	0	0	—	47	—	—	358	347
3	514	—	—	0	—	326	457	80	44
4	114	—	47	—	0	0	—	306	—
5	—	296	—	326	0	0	252	331	—
6	385	—	—	457	—	252	0	0	0
7	—	40	358	80	306	331	0	0	0
8	315	—	347	44	—	—	0	0	0

Table 2.2: The cost matrix for the internet-cable network in Example 2.4

A coalition S of some players is the collaboration of a group of cities to minimize the network payments to the internet provider. The characteristic function of this game is defined such that $c(S)$ of a coalition S is the total cost of the minimum-cost spanning tree in this subgraph $S \cup \{0\}$. Solving the GMSTP for each coalition S, we obtain the characteristic function shown in Table 2.3:

Coalition S	$c(S)$	Coalition S	$c(S)$
$\emptyset$	0	$\{L, M\}$	628
$\{K\}$	89	$\{L, N\}$	359
$\{L\}$	514	$\{M, N\}$	420
$\{M\}$	114	$\{K, L, M\}$	675
$\{N\}$	315	$\{K, L, N\}$	209
$\{K, L\}$	603	$\{K, N, M\}$	243
$\{K, M\}$	161	$\{L, N, M\}$	473
$\{K, N\}$	129	$\{K, L, N, M\}$	323

Table 2.3: The characteristic function of an Internet cable network games with four players

The least core problem is presented as follows:

$$
\begin{aligned}
\mathbf{P_4}: \quad & \min_{x,\epsilon} \quad \epsilon \\
s.t. \quad & x_K + x_L + x_M + x_N \quad = 323, \\
& x_K \leq 89 + \epsilon, \quad x_L \leq 514 + \epsilon, \quad x_N \leq 315 + \epsilon, \quad x_M \leq 114 + \epsilon, \\
& x_K + x_L \leq 603 + \epsilon, \quad x_K + x_N \leq 129 + \epsilon, \quad x_K + x_M \leq 161 + \epsilon, \\
& x_L + x_N \leq 359 + \epsilon, \quad x_L + x_M \leq 628 + \epsilon, \quad x_M + x_N \leq 420 + \epsilon, \\
& x_K + x_N + x_M \leq 243 + \epsilon, \quad x_K + x_L + x_M \leq 675 + \epsilon, \\
& x_K + x_L + x_N \leq 209 + \epsilon, \quad x_L + x_M + x_N \leq 473 + \epsilon, \\
& x_K, x_L, x_M, x_N \quad \geq 0.
\end{aligned}
\tag{2.14}
$$

In this example, the internet provider faces the GMSTP and GMSTG. Solving the GMSTP, the minimum-cost spanning tree is generated by vertices $(\{0\}, K : \{1\}, L : \{3\}, M : \{4\}, N : \{7\})$ as in Fig. 2.8. Using the previous CGA to find the core of the GMSTG, one has $\epsilon^* = 0$ (i.e., the core is non-empty) with the corresponding payoff vector $(0, 209, 114, 0)$. This means there is a stable cost allocation $x^* = (K : 0, L : 209, M : 114, N : 0)$ such that no city prefers to break the grand coalition. However, this cost allocation benefits players K and N as they have no share in the cost allocation. Using the equal profit method, we can calculate another cost allocation $x^{*\prime} = (K : 20.26, L : 117.02, M : 114, N : 71.72)$ in the GMSTG core. This cost allocation is better with respect to relative costs of the players being proportionate to the individual cost.

Using the Prim's algorithm and the Bird rule, we can find a payoff vector $(K : 89, L : 80, M : 114, N : 40)$*, which is not in the GMSTG core due to* $x_K + x_M > 161$*. However, as the optimal tree* $\tau(G)$ *has 2 branches in Fig. 2.4, we could check that the payoff vector belongs to* $\boldsymbol{Core}(\{M\}, c) \times \boldsymbol{Core}(\{K, N, L\}, c)$*. Hence, we have the strict inclusion in the Proposition 2.14.*

2.4.2 Results on larger instances

In this section, we present some computational results of the CGA to solve the GMSTG on randomly generated instances and GTSPLIB instances. The numbers of vertices for randomly generated instances range from 51 to 111 and the numbers of clusters range between 5 and 11. The inter-cluster edge costs are drawn from an uniform distribution of the range $[1, 1000]$ in a similar manner as in Golden et al. (2005).

Table 2.4 shows the results of CGA where column m is the total number of vertices in the graph and column n is the number of clusters. For each choice of (m, n) we generate 20 random instances, and each row of the table shows statistics averaged over 20 instances. In particular, *#Iter* is the number of iterations of the separation, *Ave.time per.Iter* is the average time (in seconds) per step and *CGA time* is the total time (in seconds) for solving the least core problem.

The total computation time for finding the least core is broken down into the time to solve the LP relaxation problems (column 5) and that to solve the separation step of CGA (column 6). Columns 5 and 6 show that the bottleneck is the separation step. The table also provides some comparisons between the CGA (*CGA time*) and a brute-force approach (*BF time*) for solving least core problem. For each problem configuration, column 9 shows the percentage of test instances where the least core value is greater than zero, i.e., when the core is empty. The last column of the table shows the average values of the normalized least core value $\bar{\epsilon}$ among all 20 random instances generated for each set of $\{m, n\}$. This value $\bar{\epsilon}$ is defined as the ratio of the least core value ϵ^* to the cost of the grand coalition.

Figure 2.9 shows the convergence of an experiment for a randomly generated instance with 101 vertices and 10 clusters for the least core problem using the constraint generation method, where the red and blue lines correspond to upper and lower bounds of the least core value, respectively.

From Table 2.4 and Fig. 2.9, the following observations can be made:

m	n	$\#Iter$	$Ave.time$ $per.Iter$	LP $time$	$Separation$ $time$	CGA $time$	BF $time$	$\%Empty$ $core$	$\bar{\epsilon}$
51	5	6.25	3.97	0.03	21.70	24.83	54	10	0.045
	7	10.3	4.92	0.05	45.15	50.62	228	25	0.0088
71	5	5.07	15.27	0.03	72.24	77.43	101	20	0.022
	7	9.68	19.40	0.02	159.60	187.83	436	5	0.0038
91	5	5.53	35.72	0.05	163.97	197.55	217	0	0
	7	10.55	41.76	0.07	403.72	440.57	1616	15	0.0075
61	7	9.58	9.39	0.04	80.47	89.97	1116	15	0.0019
	9	15.25	13.09	0.12	163.83	199.63	6100	30	0.0079
81	7	10.24	20.95	0.05	181.32	214.51	1434	20	0.0061
	9	15.75	29.46	0.17	431.25	463.92	6178	15	0.0074
101	7	9.58	58.57	0.04	513.87	561.06	1998	10	0.0045
	9	13.91	125.03	0.22	1563.84	1739.19	10707	10	0.0045
71	9	14.4	28.82	0.21	419.52	415.06	5395	45	0.0225
	11	22.20	30.31	0.25	774.36	672.84	25840	35	0.0086
91	9	16.40	47.73	0.19	734.47	782.80	14255	10	0.006
	11	14.40	106.88	0.09	1385.38	1539.20	22047	30	0.0082
111	9	18.25	98.73	0.28	1651.14	1801.90	34736	20	0.0055
	11	24.60	197.34	0.29	4478.99	4854.60	49038	35	0.0138

Table 2.4: Computational time and the number of iterations involved in the Constraint Generation Algorithm

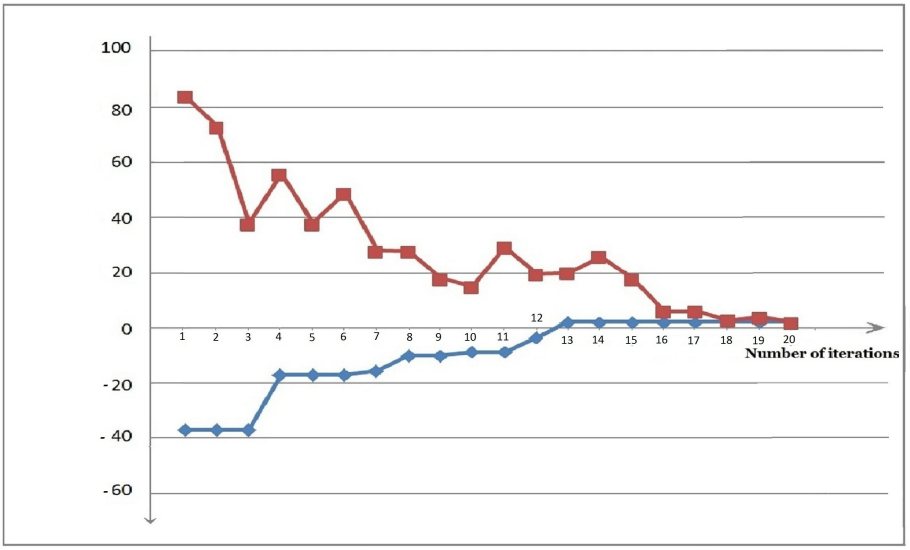

Figure 2.9: Convergence of lower and upper bounds of the least core value in the CGA for an instance with m=101 and n=10

- As we increase the number of vertices but keep the number of clusters fixed, the average time for each iteration (with the main part of the separation step) increases as we would expect.
- When the number of clusters increases, but total number of vertices remains constant, the total number of iterations for the CGA increases.
- The number of constraints needed in the final model of the CGA is significantly less than the total number 2^n of constraints required to solve problem (2.1).

In what follows, we present additional results to test the effect of the graph density on the performance of the algorithm. The results are shown in Table 2.5 where the first two columns are the number of clusters (players) and number of vertices. For these experiments, we consider 10 instances for each tuple of (m, n, δ) where δ is the density of the graph. Since we have four choices of δ, resulting in the generation of 40 random instances for each combination (m, n). Here, a density of $\delta = 0.25$ means the probability of having an inter-cluster edge between any two vertices of different clusters is 25% while a density of $\delta = 1$ means the graph is fully connected. For intra-cluster edges which is not included in the graph, we set the corresponding variables in the formulation to zero.

The computational results indicate that when we fix the instance size but increase the density, the total time for CGA increases significantly. Moreover, the core is likely to be non-empty when the edge cost matrix is less dense. The reason might be that there is fewer valid constraints in the core formulation of problem (2.1). For problem instances with the same number of players and the density, if the network graph has more vertices then the calculation time for separation problem increases, although the number of iterations in CGA does not change considerably.

For finding the least core using CGA, columns 4–6 show the computational statistics such as the average number of iterations, average time for each iteration and total time for the CGA. Column 7 presents the time required by the brute-force method to solve the least core problem. Comparing columns 6 and 7, one can see the computational advantages of CGA.

To see the algorithm's performance on more realistic instances, we performed tests on some GTSPLIB instances (Zverovich 2002) with the number of vertices $14 \leq m \leq 70$ and the number of clusters $3 \leq n \leq 14$. These instances are derived from TSPLIB instances where the vertex set has already been partitioned. However, they contain no source vertex, which is needed to model the GMSTG. For this reason, we choose the last vertex of the last non-singleton cluster to be the source vertex so that the total number of clusters in the corresponding GMSTP instance is retained. For these instances, we

m	n	δ	$\#Iter$	$Ave.time$ $per.Iter$	$Separation$ $time$	CGA $time$	BF $time$	$\%Empty$ $core$	$\bar{\epsilon}$
		0.25	1.1	9.44	2.73	10.38	62.17	0	0
51	5	0.5	1.8	7.46	5.48	13.44	42.32	0	0
		0.75	2.4	7.78	10.78	18.68	81.75	0	0
		1	4.2	5.08	19.15	21.34	58.27	10	0.0023
		0.25	1.7	7.36	3.72	12.32	125.79.3	0	0
51	7	0.5	2.5	6.11	7.16	15.28	131.65	0	0
		0.75	4.3	5.96	17.95	25.64	178.30	0	0
		1	8.7	6.68	51.17	58.15	341.62	30	0.0146
		0.25	1.0	26.29	5.24	26.29	87.53	0	0
71	5	0.5	1.7	19.0	10.61	32.3	121.65	0	0
		0.75	3.3	15.76	29.81	52.01	88.16	0	0
		1	5.4	20.86	88.32	112.67	171.49	30	0.0342
		0.25	1.2	19.95	5.21	23.84	453.12	0	0
71	7	0.5	3.9	11.57	25.39	45.14	491.63	0	0
		0.75	5.0	15.6	53.16	78.03	478.1	10	0.0031
		1	8.3	17.80	126.61	147.76	369.20	10	0.0076
		0.25	1.2	33.05	8.38	39.67	119.82	0	0
91	5	0.5	2.0	26.5	17.69	53.02	81.93	0	0
		0.75	3.1	25.16	45.95	78.01	103.19	0	0
		1	4.4	36.11	103.38	158.9	184.73	0	0
		0.25	1.2	60.65	12.07	72.78	513.63	0	0
91	7	0.5	2.4	38.02	36.82	91.24	561.16	0	0
		0.75	7.6	35.69	210.34	271.27	631.35	10	0.0022
		1	7.9	50.37	309.89	397.92	1826.79	20	0.0149

Table 2.5: Computation time of the least core with instances of different density.

also calculate the total cost of the grand coalition $c(N)$ to compute the normalized least core values. The results are presented in Table 2.6.

The results presented in Table 2.6 indicate that when the number of vertices is less than 71, the computational time of CGA speeds up as the number of players increases to 14. Indeed, the performance of the algorithm on GMSTG least core problem depends on both number of vertices and number of clusters. Combining all numerical results, it can be clearly seen that the CGA outperforms the brute-force method in terms of the computational time.

The only case of "11berlin52", where the core is empty, constitutes less than 10% of all the instances in Table 2.6. This result parallels those shown in Table 2.4 and 2.5, in which the $\%Empty\ core$ column shows that the core is empty in a relatively small percentage of all cases. Another general observation is that in case of empty-core instances, the

Problem name	m	n	*#Iter*	*Separation time*	*CGA time*	*BF time*	$c(N)$	$\bar{\epsilon}$
3burma14	14	3	1	0.06	0.31	0.45	9.2	0
4ulysses16	16	4	3	0.20	0.45	0.75	25.49	0
4gr17	17	4	2	0.21	0.51	0.84	736	0
5gr21	21	5	6	0.68	0.87	1.25	1160	0
5ulysses22	22	5	4	0.81	1.09	2.69	33.76	0
5gr24	24	5	4	1.20	1.61	2.38	259	0
6fri26	26	6	10	4.46	5.3	6.29	388	0
10att48	48	10	63	316.64	325.42	554.43	12725	0
11eil51	51	11	89	750.44	761.24	1240.7	134.9	0
11berlin52	52	11	80	717.96	730.21	1891.4	2938.4	0.00045
14st70	70	14	136	6210.23	6245.9	36713	243.1	0

Table 2.6: Computation time of the least core with instances of different density.

least core value is less than 5% of the cost of the grand coalition. The computational results presented in Table $4, 5, 6$ collectively suggest a high probability of the existence of the core in GMSTGs, which is related to the theoretical property of the MSTG in which the core is always non-empty.

2.5 Conclusion and Future Work

In this chapter, we have introduced a new class of cooperative games, namely the Generalized Minimum Spanning Tree Game, and proposed a constraint generation method to solve the least core problem. We have provided an example showing an empty core for the GMSTG. Numerical results have shown that randomly generated instances defined on complete graphs with up to 111 vertices and 11 clusters can be solved to optimality using the proposed method. We also tested some GTSPLIB instances with up to 70 vertices and with up to 14 clusters and observed the clear advantage of using the constraint generation algorithm. This chapter has been published in Le et al. (2016) as one of its research output. By introducing the new generalized minimum spanning tree game, its applications and an algorithm for calculating a stable cost share, we hope the new game will help to promote further practical usage of cooperative game theory in cost allocation problems.

Chapter 3

The Shapley value of Linear Production Games

3.1 Introduction

The Linear Production Game (LPG) is a model of collaboration arising in a production environment in which several independent decision makers are involved to make a joint production maximisation based on their combined resources. Owen (1975) describes the LPG as a type of cooperative game with transferable utilities (TU) that provides for a framework to understand the cooperation among the resource owners (players) in such a context. The players work jointly to produce finished goods which can be sold at a given market price, and wish to find a fair way to share the payoff of the collaboration. Through working in the collaboration, the team can manufacture more products, and hence, can collectively generate more profit to the members.

Linear Production Games have been widely studied over the past 40 years. Granot (1986) generalises Owen's LPG model without the additivity assumption of the resource function. Variations of this model include minimum cost spanning tree games (Granot & Huberman 1981), assignment games (Shapley & Shubik 1971) and network synthesis games (Tamir 1991). Some other extensions of linear production games are investigated in the works of Fernández et al. (2005) and Lozano (2013). Recently, Nishizaki et al. (2016) examined a linear production game with restricted communication represented by a network, they then developed the extended Owen solution and showed its geometric properties.

One of the fundamental questions in cooperative game theory is how to share the payoff/cost in a fair way among the players. Several solutions concepts have been proposed

to address this issue, which include the core, the least core, the Shapley value and the Banzhaf index. Among these, the Shapley value is one of the most popular, particularly in cooperative games with transferable utility, probably due to its relevant economics interpretation on marginal contributions and its desirable mathematical property, i.e., it is the unique concept that satisfies four axioms including efficiency, symmetry, dummy, and additivity. Not only the Shapley value is interpreted as the payoff/cost distribution but also it suggests a measure of the power of the players in a voting game (Shapley & Shubik 1954, Bilbao et al. 2002). In network applications, the Shapley value has been employed as a measure of centrality to identify the important nodes (Gomezb et al. 2003, Michalak et al. 2013).

In general, one cannot calculate the Shapley value in polynomial time for many coalition games due to the fact that its formulation involve an exponentially large number of terms. This computational difficulty also hold for structured games such as weighted voting games and minimum cost spanning tree games (see, Deng & Papadimitriou 1994, Ando 2010, for more detail). In general, finding the exact Shapley Value for large-scale cooperative games, including Linear Production Games is computationally intractable. In practice, we have observed this for the games with more than 30 players.

In this chapter, we propose a method combining linear programming sensitivity analysis and a stratified sampling technique to compute the Shapley value of Linear Production Games. Since we have to compute a large number of marginal contributions in the Shapley value, the linear programming sensitivity analysis is utilised to improve the calculation time of these marginal contributions. In addition, a new randomised algorithm using Monte Carlo simulation is constructed to help restricting the number of marginal contributions we will evaluate in LPGs with large number of players. Therefore, the main contributions of this chapter include:

- We derive an alternative closed form solution for calculating the exact Shapley value of a special class of Linear Production Games (in Section 3.3.1)
- We present a new method to improve the computation of the Shapley value utilising linear programming sensitivity analysis (in Section 3.3.2).
- For large-scale Linear Production Games, we additionally use a stratified sampling technique to approximate the Shapley value (in Section 3.3.3).
- Finally, we provide numerical results to illustrate the effectiveness of the proposed methods compared to existing methods (in Section 3.4).

3.2 Preliminaries and Notation

In this section, we first provide a formal definition of the Linear Production Game (LPG), following which we define the Shapley value and its formulation.

3.2.1 Linear Production Game (LPG)

Consider a set N of n players who own a set R of r resources available for the production of a set P of p products. Each player i has a vector of resources $\boldsymbol{b}_i = (b_i^1, b_i^2, \ldots, b_i^r)^T$, i.e., that player has exactly b_i^k unit of resource $k \in \{1, 2, \ldots, r\}$. Manufacturing product $j \in \{1, 2, \ldots, p\}$ requires a_{kj} units of resource k, for $k \in \{1, 2, \ldots, r\}$. Product j can be sold in the market at price c_j.

For any arbitrary coalition S of players in N, the combined resource vector is $\boldsymbol{b}(S) = (b^1(S), b^2(S), \ldots, b^r(S))^T = \sum_{i \in S} \boldsymbol{b}_i$, where $b^k(S) = \sum_{i \in S} b_i^k$, for $\forall k \in \{1, 2, \ldots, r\}$. The value of a coalition S is the maximum payoff this group can achieve with all the resources possessed by its members. Let the vector of production output be denoted as $\boldsymbol{x} = (x_1, \ldots, x_p)$ with x_p is the amount of product p manufactured, then the payoff $\nu(S)$ assigned to a coalition S is defined as the total profit that the coalition can collectively gained. Hence, $\nu(S)$ can be obtained by solving the following linear program:

$$\begin{aligned} \nu(S) := \quad \max \quad & \sum_{j=1}^{p} c_j x_j \\ \text{s.t.} \quad & \sum_{j=1}^{p} a_{kj} x_j \leq b^k(S), \quad \forall k = \{1, 2, \ldots, r\}, \\ & x_j \geq 0, \quad \forall j = \{1, 2, \ldots, p\}, \end{aligned} \tag{3.1}$$

or equivalently, $\nu(S) := \max \{\boldsymbol{c}^T \boldsymbol{x} \mid \mathcal{A}\boldsymbol{x} \leq \boldsymbol{b}(S),\ \boldsymbol{x} \geq 0\}$, where $A = \{a_{kj}\}_{k=1,\ldots,r}^{j=1,\ldots,p}$ is the resource requirement matrix.

Owen (1975) shows that the LPG is a totally balanced game, i.e. the LPG core is always non-empty for all its induced subgame. and the characteristic function is super-additive; that is for any two disjoint coalitions S_1 and S_2, we have $\nu(S_1) + \nu(S_2) \leq \nu(S_1 \cup S_2)$. Therefore, the grand coalition will be formed for the benefit of all players. The LPG game is concerned with dividing the total payoff gained by the grand coalition to its players.

3.2.2 Shapley Value

The Shapley value (Shapley 1953) is among the most popular solution concepts in cooperative game theory for dividing the total payoff/cost to the players, assuming that they all collaborate. In the concept, the allocation for a player i is proportional to that player's contribution to the game, i.e., how much value, player i creates. The formula of the Shapley value of player i is the weighted average of all the marginal contributions of that player to all coalitions $S \subseteq N \setminus \{i\}$.

Formally, the *Shapley value* $\boldsymbol{\phi} \in \mathbb{R}^n$ of a cooperative game $G = (N, \nu)$ is the allocation of payoff where

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(n - |S| - 1)!}{n!} [\nu(S \cup \{i\}) - \nu(S)], \tag{3.2}$$

where we use the short-hand notation $\nu(S \cup \{i\}) - \nu(S)$ to denote the marginal contribution of player i to the coalition S.

Another way to formulate the Shapley value is as follows. Suppose the ordering of the arrival of players $\pi : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$ is represented as a permutation function, where $\pi(j) = i$ means that player i joins the group in position j. Denote $\Pi(N)$ as the set of all permutations $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$ of players in N, hence $|\Pi(N)| = n!$. For each π and $i = \pi(j)$, let $\text{Pre}^i(\pi) := \{\pi(1), \pi(2), \ldots, \pi(j-1)\}$ be the set of players who are the predecessors of player i in π. Given a permutation π ordering the arrival of the players, the marginal contribution of player $i = \pi(j)$ is the difference $\nu(\pi(1), \ldots, \pi(j)) - \nu(\pi(1), \ldots, \pi(j-1))$. The Shapley value ϕ_i of a player i is the average over all permutations $\pi \in \Pi(N)$, of the marginal contribution of the player i to the set of players who arrive before player i. Therefore, the value is calculated as:

$$\phi_i = \frac{1}{n!} \sum_{\pi \in \Pi(N)} [\nu(\text{Pre}^i(\pi) \cup \{i\}) - \nu(\text{Pre}^i(\pi))]. \tag{3.3}$$

In super-additive games such as LPG, the condition $\nu(i) \leq \nu(\text{Pre}^i(\pi) \cup \{i\}) - \nu(\text{Pre}^i(\pi))$ implies $\nu(i) \leq \phi_i$, i.e., the payoff allocation $\boldsymbol{\phi}$ satisfies the individual rationality property, making the Shapley value also an imputation allocation. The Shapley value defined in (3.2) is shown to be the unique division scheme that meets four desirable criteria (axioms):

- Efficiency: $\sum_{i \in N} \phi_i(G) = \nu(N)$;
- Null player: If $\nu(S \cup \{i\}) = \nu(S), \forall S \subset N$, then $\phi_i(G) = 0$;

- Symmetry: If players i and j are symmetry, i.e., $\nu(S \cup \{i\}) = \nu(S \cup \{j\}), \forall S \subset N$ and $i, j \notin S$, then $\phi_i(G) = \phi_j(G)$;
- Additivity: For any games $G^1 = (N, \nu^1)$ and $G^2 = (N, \nu^2)$, the sum game $G^1 \oplus G^2 := (N, \nu^1 + \nu^2)$ satisfies $\phi_i(G^1 \oplus G^2) = \phi_i(G^1) + \phi_i(G^2)$ for all $i \in N$.

The first axiom requires that players divide all the payoff/cost available to the grand coalition among themselves. The second axiom requires that zero payoff is assign to the player who does not contribute to any coalition. In the third axiom, two players i, j are said to be symmetric in the game v if they make the same marginal contributions to any coalition. The axiom demands to pay symmetric players equal shares. Finally, the payoff of a player in the game $G^1 \oplus G^2$ should be equal to the sum of that player's payoffs in two separate games G^1 and G^2.

Another special property of Shapley value for a fair payoff allocation is the monotonicity (Young 1985). This states that if we change a game such that the contributions of a player to all coalitions stays the same or increases then the final payoff for that player should not decease. The author also show that the Shapley value can be described simply by the three properties of efficiency, symmetry, and monotonicity for all cooperative games.

3.3 Computational Methods for Finding the Shapley Value

In this section, we first provide analytical properties of the Shapley value of the LPGs by exploiting the dual structure of the characteristic function. We then apply linear programming sensitivity analysis (LPSA) to provide a fast computation of the marginal contribution of a player i to coalition S. Finally, since solving 2^n LP formulations to compute $\nu(S)$ for all possible $S \subseteq N$ could be time-consuming, we propose a randomised algorithm combining LPSA and stratified sampling technique to approximate the Shapley value.

3.3.1 Analytical Properties of the Shapley Value in LPGs

We first observe that, in reality, while the number of players might change, we often have a reasonable small and fixed number of resource types. We are interested in how the complexity of computing the Shapley value grows with the increase of the number of players. First, we formulate the dual of the characteristic function $\nu(S)$ for each coalition S:

$$\nu(S) := \min_{\boldsymbol{y} \in Y} \ \boldsymbol{b}(S)^T \boldsymbol{y}, \tag{3.4}$$

where $Y = \{\boldsymbol{y} \in \mathbb{R}^r \ : \ \mathcal{A}^T \boldsymbol{y} \geq c, \boldsymbol{y} \geq 0\}$ is the feasible space of the dual problem. It is interesting to note that this feasible set is independent on the set of players.

Let $\boldsymbol{y}_i^*, \ i = 1, \ldots, n$, be an optimal basic feasible solution of the dual problem (3.4) with the objective coefficients $\boldsymbol{b}_i$. For simplicity, we assume that the input data of the problem is chosen such that no coalition S would have multiple optimal solutions in this problem of computing $\nu(S)$. The presumption can be achieved by slightly perturbing the data of the LPG. We note that similar results derived in this work still applies even without this assumption, however, we need to be more careful in some of the statements.

We can see that if $\boldsymbol{b}_i$ are all equal then, by symmetry, $\boldsymbol{y}_i^* = \boldsymbol{y}_j^*, \ \forall 1 \leq i < j \leq n$, and the Shapley value of all players should be the same and is equal to $\boldsymbol{b}_i^T \boldsymbol{y}_i^*$. It is noted, however, that the resource vectors $\boldsymbol{b}_i, i = 1, \ldots, n$, do not have to be exactly the same for $\boldsymbol{y}_i^* = \boldsymbol{y}_j^*, \ \forall 1 \leq i < j \leq n$. Instead, we show a condition on $\boldsymbol{b}_i, i = 1, \ldots, n$, for this to hold. Let us first provide some additional notation.

Let $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_p$ be the columns of $\mathcal{A}$. For each $\boldsymbol{y} \in Y$, let $I(\boldsymbol{y}) \subseteq \{1, \ldots, p\}$ be the set of active constraints at $\boldsymbol{y}$, i.e., $\boldsymbol{a}_j^T \boldsymbol{y} = c_j, \ \forall j \in I(\boldsymbol{y})$. We denote $\mathcal{A}_{I(\boldsymbol{y})}$ as the set of active (tight) rows of $\mathcal{A}^T$ at $\boldsymbol{y}$. For each $i = 1, \ldots, n$, let us define

$$\mathcal{K}_i = \left\{ - \sum_{q \in I(\boldsymbol{y}_i^*)} \gamma_q \boldsymbol{a}_q \ : \ \gamma_q \geq 0 \right\},$$

as the cone formed by the negative of the tight rows in $\mathcal{A}_{I(\boldsymbol{y}_i^*)}$. Therefore, if the resource vector of a coalition S satisfies the condition that $-\boldsymbol{b}$ belong to the cone $\mathcal{K}_i$, the optimal dual solution of $\nu(S)$ in (3.4) is exactly $\boldsymbol{y}_i^*$. We can also show that $(\mathcal{K}_i - \boldsymbol{y}_i^*)$ is indeed the polar cone of the support cone of $(Y - \boldsymbol{y}_i^*)$ at $\mathbf{0}$.

We have the following results on the closed form solution of the Shapley value if the LPG has some special properties.

Proposition 3.1. *If $-\boldsymbol{b}_i \in \mathcal{K}_1$ for all $i = 2, \ldots, n$, then the following results hold:*

(a) There exists vector $\boldsymbol{y}^*$ such that $\boldsymbol{y}_i^* = \boldsymbol{y}^*, \ \forall \ i = 1, \ldots, n$.

(b) For all $S \subseteq N$, $\boldsymbol{y}^* \in \operatorname{argmin}_{\boldsymbol{y} \in Y} \ \boldsymbol{b}(S)^T \boldsymbol{y}$.

(c) The characteristic function ν of the LPG is additive and the closed form solution of the Shapley value is given by $\phi_i = \boldsymbol{b}_i^T \boldsymbol{y}^*$.

Proof.

(a) The proof of this part is straightforward since as long as the negative of the objective vector $-\boldsymbol{b}_i$ belongs to the cone $\mathcal{K}_1$, then the corresponding optimal dual solution $\boldsymbol{y}_i^* = \boldsymbol{y}_1^*$.

(b) Since $\boldsymbol{y}_i^* = \boldsymbol{y}_j^*,\ \forall 1 \leq i < j \leq n$, we have $\mathcal{K}_i = \mathcal{K},\ \forall 1 \leq i \leq n$ for some cone $\mathcal{K}$. By the definition of $\boldsymbol{y}_i^*$, i.e., $\boldsymbol{y}_i^* \in \underset{\boldsymbol{y} \in Y}{\operatorname{argmin}}\ \boldsymbol{b}_i^T \boldsymbol{y}$, and by the definition of $\mathcal{K}_i$, we have $-\boldsymbol{b}_i \in \mathcal{K}_i \equiv \mathcal{K}$. Therefore $-\boldsymbol{b}(S) = \sum_{i \in S}(-\boldsymbol{b}_i) \in \mathcal{K}$ and hence $\boldsymbol{y}^* \in \underset{\boldsymbol{y} \in Y}{\operatorname{argmin}}\ \boldsymbol{b}(S)^T \boldsymbol{y}$.

(c) From part (b) we have

$$\nu(S) = \boldsymbol{b}(S)^T \boldsymbol{y}^* = \sum_{i \in S} \boldsymbol{b}_i^T \boldsymbol{y}^* = \sum_{i \in S} \nu(\{i\}).$$

Thus, the characteristic function v of the LPG is additive and $\nu(S \cup \{i\}) - \nu(S) = \nu(\{i\})$. We substitute into the Shapley formula and have the closed form solution as $\phi_i = \boldsymbol{b}_i^T \boldsymbol{y}^*$. the closed form solution of the Shapley value is given by $\phi_i = \boldsymbol{b}_i^T \boldsymbol{y}^*$.

□

Proposition 3.1 is useful when the resource vectors $\boldsymbol{b}_i, i = 1, \ldots, n$, are reasonably close to each other in directions such that they all belongs to $-\mathcal{K}_1$. In the case these resource vectors spread out such that $\boldsymbol{y}_i^* \neq \boldsymbol{y}_j^*$ for some $1 \leq i < j \leq n$, the result no longer holds and we need to work with the extreme points of Y which is fixed and independent on the set of players.

Let $\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, \ldots, \boldsymbol{y}^{(\Gamma)}$ be the extreme points of Y. Each coalition S corresponds to an objective vector $\boldsymbol{b}(S)$ and a corresponding optimal solution $\boldsymbol{y}_S^*$ that belongs to the set $\{\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, \ldots, \boldsymbol{y}^{(\Gamma)}\}$. The corresponding coalition value is $\nu(S) = \boldsymbol{b}(S)^T \boldsymbol{y}_S^*$.

Conversely, let $\mathrm{H}_q, q = 1, \ldots, \Gamma$, be the set of coalitions whose optimal dual variable coincides with $\boldsymbol{y}^{(q)}$, i.e.,

$$\mathrm{H}_q = \left\{ S \subseteq N \ :\ \boldsymbol{y}^{(q)} \in \underset{\boldsymbol{y} \in Y}{\operatorname{argmin}}\ \boldsymbol{b}(S)^T \boldsymbol{y} \right\}.$$

Let $\gamma_k = \dfrac{k!(n-k-1)!}{n!}$, for $k \in \{0, 1, \ldots, n-1\}$. We can rewrite the Shapley value formulation as follows:

$$\begin{aligned}
\phi_i &= \sum_{k=0}^{n-1} \sum_{|S|=k} \gamma_k \left[\nu(S \cup \{i\}) - \nu(S)\right] \\
&= \sum_{k=0}^{n-1} \sum_{q_1=1}^{\Gamma} \sum_{q_2=1}^{\Gamma} \left\{ \sum_{|S|=k,\ S \in \mathrm{H}_{q_1},\ S\cup\{i\} \in \mathrm{H}_{q_2}} \gamma_k \left[\nu(S \cup \{i\}) - \nu(S)\right] \right\} \\
&= \sum_{k=0}^{n-1} \sum_{q_1=1}^{\Gamma} \sum_{q_2=1}^{\Gamma} \left\{ \sum_{|S|=k,\ S \in \mathrm{H}_{q_1},\ S\cup\{i\} \in \mathrm{H}_{q_2}} \gamma_k \left[\boldsymbol{b}(S \cup \{i\})^T \boldsymbol{y}^{(q_1)} - \boldsymbol{b}(S)^T \boldsymbol{y}^{(q_2)}\right] \right\} \quad (3.5)
\end{aligned}$$

Remark 3.2. The alternative analytical formulation for the Shapley value in Equation (3.5) is useful for cases when the numbers of resources and products (r, p) are relatively small compared to the number of players n. In particular, we show an example when the number of extreme points of Y is relatively small compared to the total number of coalitions.

Consider, for example, the case of $n = 30$, $p = 3$ and $r = 2$. Using the Formulation (3.1) would involve solving $2^n \approx 10^9$ LPs, each with p decision variables and with $(r + p)$ constraints. If we use Equation (3.5), we notice that Y has at most $\binom{r+p}{p} = 10$ extreme points and the equation involves only $10^2 = 100$ pairs of (q_1, q_2). In each pair, we only need to evaluates linear functions on those coalitions that satisfy $\{|S| = k, S \in \mathrm{H}_{q_1}, S \cup \{i\} \in \mathrm{H}_{q_2}\}$. Let coalition S be rewritten as a binary vector $\boldsymbol{z}(S) \in \{0, 1\}^n$ with $z_i = 1$ if and only if player i is in the coalition. We also denote $\boldsymbol{b}(S) = \mathcal{B}\boldsymbol{z}(S)$ where $\mathcal{B}$ is a matrix of resources vectors in the columns.

The following remark helps speed up the process of finding the optimal solution for each coalition S by checking a set of linear constraints on $\boldsymbol{z}(S)$.

Proposition 3.3. *The condition of $S \in \mathrm{H}_q$ for $q \in \{1, \ldots, \Gamma\}$ is equivalent to:*

- *If the vertex $\boldsymbol{y}^{(q)}$ is non-degenerate, $(\mathcal{A}_{I(\boldsymbol{y}^{(q)})})^{-1}\mathcal{B}\boldsymbol{z}(S) \geq 0$, which is a set of linear constraints on $\boldsymbol{z}(S)$.*
- *If the vertex $\boldsymbol{y}^{(q)}$ is degenerate, $(\mathcal{A}_I^{(q)})^{-1}\mathcal{B}\boldsymbol{z}(S) \geq 0$, $\forall \mathcal{A}_I^{(q)} \subset \mathcal{A}_{I(\boldsymbol{y}^{(q)})}$. In particular, when $\mathcal{A}_{I(\boldsymbol{y}^{(q)})}$ is not a square matrix, we can list out all combinations of columns of $\mathcal{A}_I^{(q)} \subset \mathcal{A}_{I(\boldsymbol{y}^{(q)})}$ such that each set of columns corresponds to a basis.*

Proof. We give a sketch proof here for the proposition. More detailed explanation can be found in chapter 4 of the book by Bertsimas & Tsitsiklis (1997).

The condition $S \in \mathrm{H}_q$ is equivalent to $\boldsymbol{y}^{(q)} \in \underset{\boldsymbol{y} \in Y}{\operatorname{argmin}}\ \boldsymbol{b}(S)^T \boldsymbol{y}$. That means the dual problem (3.4) has $\boldsymbol{y}^{(q)}$ as one of its optimal solution, hence, both conditions of feasibility

and optimality must be satisfied. The feasibility condition of the vertex in the dual polyhedron is straightforward, however, we also need to check the optimality condition, which is the same as the feasibility condition of the primal problem. Therefore, if a basis B is optimal in the primal problem, the condition of $S \in \mathrm{H}_q$ is equivalent to $B^{-1}\boldsymbol{b}(S) = (\mathcal{A}_{I(\boldsymbol{y}^{(q)})})^{-1}\mathcal{B}\boldsymbol{z}(S) \geq 0$ for the non-degenerate vertex $\boldsymbol{y}^{(q)}$.

□

Remark 3.4. In the case of the polyhedron of the dual problem (3.4) has a small number of vertices and there are a large number of game players, we can efficiently find the optimal solution for each coalition S by checking which value of q satisfies $S \in \mathrm{H}_q$. This step help us in the process of finding the Shapley value quickly.

Example 3.1. *We give the detail of a game illustration with $r = 2$ resources and $p = 3$ products to show the insight of this remark*

$$\begin{aligned} \textit{Primal}\ (\boldsymbol{b}) := \quad \max \quad & 3x_1 + 8x_2 + x_3 & \\ s.t. \quad & 5x_1 + 19x_2 + 11x_3 & \leq b_1, \\ & 13x_1 + 17x_2 + 23x_3 & \leq b_2, \\ & x_1, x_2, x_3 & \in \mathbb{R}_+, \end{aligned} \tag{3.6}$$

The dual problem and its feasible area are shown as following:

$$\begin{aligned} \textit{Dual}\ (\boldsymbol{b}) := \quad \min \quad & b_1y_1 + b_2y_2 & \\ s.t. \quad & 5y_1 + 13y_2 & \geq 3, \\ & 19y_1 + 17y_2 & \geq 8, \\ & 11y_1 + 23y_2 & \geq 1, \\ & y_1, y_2 & \in \mathbb{R}_+, \end{aligned} \tag{3.7}$$

Suppose that there are only two types of players in our game, i.e. $m = 12$ players with the resource vector $\boldsymbol{b} = (50, 37)$ and $m' = 18$ players with the resource vector $\boldsymbol{b}' = (28, 371)$. The feasible area of the dual problem has three basic feasible solutions at three extreme points $A = (0, 0.47), B = (0.33, 0.1), C = (0.6, 0)$.

Following the arguments of remark 3.2, we have the resource constraint matrix

$$\mathcal{A} = \begin{pmatrix} 5 & 19 & 11 & 1 & 0 \\ 13 & 17 & 23 & 0 & 1 \end{pmatrix},$$

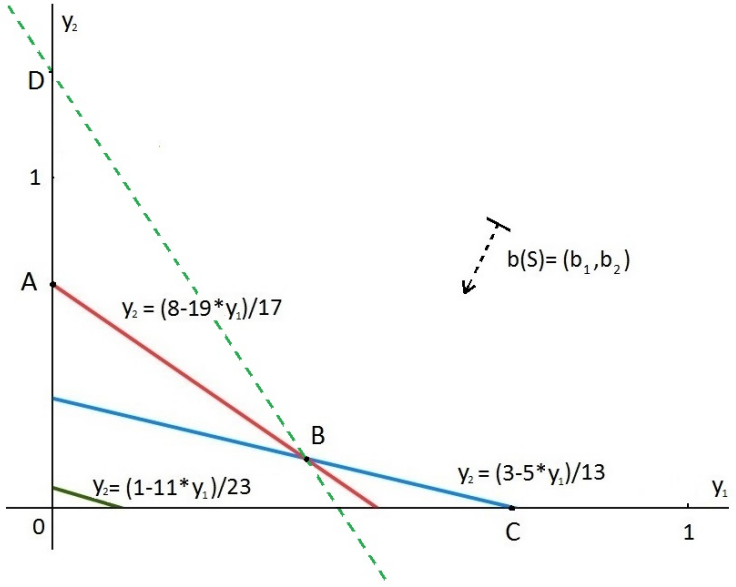

Figure 3.1: Dual space of the example with the resource vector $b(S)$

from which we compute the set of active constraints and active rows at each vertices $\boldsymbol{y}_A, \boldsymbol{y}_B, \boldsymbol{y}_C \in Y$. *In particular, we have* $I(\boldsymbol{y}_A) = \{2,4\}, I(\boldsymbol{y}_B) = \{1,2\}$, *and* $I(\boldsymbol{y}_C) = \{1,5\}$; *with* $\mathcal{A}_{I(\boldsymbol{y}_A)} = \begin{pmatrix} 19 & 1 \\ 17 & 0 \end{pmatrix}$, $\mathcal{A}_{I(\boldsymbol{y}_B)} = \begin{pmatrix} 5 & 19 \\ 13 & 17 \end{pmatrix}$, *and* $\mathcal{A}_{I(\boldsymbol{y}_C)} = \begin{pmatrix} 5 & 0 \\ 13 & 1 \end{pmatrix}$.

For each coalition S *of the game, we need to check which* $q \in \{A, B, C\}$ *s.t.* $S \in \mathrm{H}_q$. *For the two resource vectors* $\boldsymbol{b} = (50, 37)$ *and* $\boldsymbol{b}' = (28, 371)$, *the player type* $\boldsymbol{b}$ *belongs to* H_A *and the player type* $\boldsymbol{b}'$ *belongs to* H_C. *Now, considering a coalition* S^* *of eight players of type* $\boldsymbol{b}$ *and one player of type* $\boldsymbol{b}'$, *i.e. the total resource vector of the coalition is* $\boldsymbol{b}(S^*) = (428; 667)$. *Among the three vertices represent non-degenerate basic feasible solutions, we can check that the only satisfied inequalities* $inv(\mathcal{A}_{I(\boldsymbol{y}_B)})\boldsymbol{b}(S^*) \geq 0$ *indicates* $S^* \in \mathrm{H}_B$, *i.e.* B *is the optimal vertex for the dual problem.*

To demonstrate the effect of degeneracy on the dual and how we could overcome this issue, we suppose that there is another extra constraint $14y_1 + 4y_2 \geq 5$ *in the dual problem (3.7). The new feasible space has three vertices as* $\{D, B, C\}$ *where* $D = (0, 1.25)$. *We have matrix* $\mathcal{A} = \begin{pmatrix} 14 & 5 & 19 & 11 & 1 & 0 \\ 4 & 13 & 17 & 23 & 0 & 1 \end{pmatrix}$, *and the sets of active constraints are* $I(\boldsymbol{y}_C) = \{2,6\}, I(\boldsymbol{y}_B) = \{1,2,3\}$, *and* $I(\boldsymbol{y}_D) = \{1,5\}$.

In this case, the vertex B *becomes degenerate with* $\mathcal{A}_{I(\boldsymbol{y}_B)} = \begin{pmatrix} 14 & 5 & 19 \\ 4 & 13 & 17 \end{pmatrix}$. *We check that the second criteria of the Proposition 3.3 satisfies for all basic representations* $\mathcal{A}_I^{(B)} = \left\{ \begin{pmatrix} 14 & 5 \\ 4 & 13 \end{pmatrix}, \begin{pmatrix} 5 & 19 \\ 13 & 17 \end{pmatrix}, \begin{pmatrix} 19 & 14 \\ 17 & 4 \end{pmatrix} \right\}$ *of the vertex* B. *Since*

$(\mathcal{A}_I^{(B)})^{-1}\mathcal{B}\boldsymbol{z}(S) \geq 0$ *for all representations, this confirm that* B *is still the optimal vertex of the modified problem.*

3.3.2 Linear Programming Sensitivity Analysis (LPSA)

The motivation of using LPSA is based on the following observation. Adding a player i to a coalition S of other players, in many cases, only slightly modifies the resource vector of the coalition. Sensitivity analysis dictates that the right-hand side values of the Linear Programming model can vary within certain limits without causing a change in the optimal solution. The LPSA is therefore used to calculate $\nu(S \cup \{i\})$ and hence the marginal contribution value $\nu(S \cup \{i\}) - \nu(S)$, instead of the need to re-evaluate.

The LPSA is applied to calculate the marginal contribution $\nu(S \cup \{i\}) - \nu(S)$ for each coalition S as follows. First, suppose the calculation of $\nu(S)$ can be done by solving the following standard LP:

$$(\mathbf{P}:) \quad \nu(S) = \min\{\boldsymbol{f}^T\boldsymbol{\omega} \mid \mathrm{D}\boldsymbol{\omega} = \boldsymbol{b}(S),\ \boldsymbol{\omega} \geq 0\},$$

where $\boldsymbol{f}$, D are coefficients derived from $(\boldsymbol{c},\ \mathcal{A})$ after transforming Problem (3.1) into the standard form (with slack variables added).

Using a simplex method in Tabular Form, a basis B is optimal if the following two conditions hold (i) Feasibility: $B^{-1}\boldsymbol{b}(S) \geq 0$ and (ii) Optimality: $\boldsymbol{f}^T - \boldsymbol{f}_B^T B^{-1}\mathrm{D} \geq 0^T$.

To calculate $\nu(S \cup \{i\})$ in detail, let (**P'**) denotes the new LP where the right hand side coefficients $\boldsymbol{b} := \boldsymbol{b}(S) = \{b^k(S)\}_{k=1}^r$ are changed to $\boldsymbol{b} + \boldsymbol{\Delta} := \boldsymbol{b}(S \cup \{i\}) = \boldsymbol{b}(S) + \boldsymbol{b}_i = \{\boldsymbol{b}^k(S) + \Delta^k\}_{k=1}^r$, where $\boldsymbol{\Delta} := \{\Delta^k\}_{k=1}^r = \{b_i^k\}_{k=1}^r$. The optimality condition of (**P'**) is still satisfied, as in (**P**). However the new feasibility condition is $B^{-1}(\boldsymbol{b} + \boldsymbol{\Delta}) \geq 0$ may not hold. We can divide the situation into two cases:

- If the resource vector $\boldsymbol{\Delta} = \boldsymbol{b}^i$ satisfies the new feasibility condition then the optimal basis of the problem (**P'**) is still the same as in (**P**). Hence, there is a closed form outcome of the problem (**P'**) as $\nu(S \cup \{i\}) = \boldsymbol{f}_B B^{-1}(\boldsymbol{b} + \boldsymbol{\Delta})$.
- Otherwise, then the optimal solution and the basis of (**P**) are used as the starting point for the dual simplex algorithm to solve (**P'**) (see Wolsey & Nemhauser 1999, for detail about the dual simplex method).

3.3.3 LPSA Randomized Algorithm

In this section, we describe a stratified sampling method for approximating the Shapley value instead of solving all 2^n coalition values. First, we rewrite the Shapley value as follows:

$$\begin{aligned}\phi_i &= \frac{1}{n!}\sum_{\pi\in\Pi(N)}[\nu(\mathrm{Pre}^i(\pi)\cup\{i\})-\nu(\mathrm{Pre}^i(\pi))].\\ &= \sum_{S\subseteq N\setminus\{i\}}\frac{|S|!(n-|S|-1)!}{n!}[\nu(S\cup\{i\})-\nu(S)]\\ &= \frac{1}{n}\sum_{k=0}^{n-1}\underbrace{\frac{1}{\binom{n-1}{k}}\sum_{|S|=k,\ i\notin S}[\nu(S\cup\{i\})-\nu(S)]}_{\mathbb{E}(X_i^k)},\end{aligned} \tag{3.8}$$

where $\mathbb{E}(X_i^k)$ is the average of all marginal contributions of player i to coalitions of size k. Formulation (3.8) suggests the use of a Monte Carlo simulation approach based on stratified sampling. This Monte Carlo method was first applied in Mann & Shapley (1960) to approximate the Shapley value, and then analyse the Electoral College vote system in the UP Presidential elections.

In the sampling model of the Shapley value for player i, the population contains all marginal contribution $\nu(S\cup\{i\})-\nu(S)$, where S is a subset of $N\setminus\{i\}$. It can be divided into n separate strata $\{M_k\}_{k=0}^{n-1}$ correspondingly, where stratum M_k contains all $k!(n-k-1)!$ copies of each coalition of size k . The total number of samples is allocated to n stratum by the proportional allocation rule, more detail can be found in the Appendix A.2.3. For each strata, we take the simple random sampling to approximate the expected value $\mathbb{E}(X_i^k)$, and the last equation of (3.8) is applied for the final calculation.

We now propose a new randomised method to take samples which will strengthen the effect of linear programming sensitivity analysis. Most approaches in the literature do sampling separately for each player. After taking a sample coalition S, we choose all players $i\notin S$ one by one to form the bigger coalition $S\cup\{i\}$. The process saves $(n-|S|-1)$ times of recalculating $\nu(S\cup\{i\})$ for all different player $i\notin S$. Afterwards, instead of finding the marginal contribution of that player i to coalition S directly, we utilise the Linear Programming sensitivity analysis to speed up the process.

Finally, we describe the full LPSA randomised scheme in Algorithm 2.

Here we apply the simple random method into the process of sampling from each stratum. The LPSA Randomized algorithm estimates the stratum mean $\mu_i^k=\mathbb{E}(X_i^k)$ based on sample average $\overline{X_i^k}$. Suppose that the number of samples taken from each stratum is sufficiently large so that we can use the central limit theorem to approximate the distribution of X_i^k by a Gaussian with mean μ_i^k and the standard deviation σ_i^k. From the

Algorithm 2 Shapley value - LPSA Randomized Algorithm

Input: The problem instance size (p, r, n) and LPG data $(\mathcal{A}, B, c)$
Output: Approximated Shapley value for each player i.
Initialize: Divide total m samples into $\{m_k\}_{k=0}^{n-1}$ strata by proportional allocation
# Where m_k is the sample size of stratum M_k;
for $k = 0 : (n-1)$ **do**
  if the stratum population $\binom{n-1}{k} < m_k$ **then**
    Compute $\nu(S \cup \{i\}) - \nu(S)$ of all $S \in M_k$
    # Finding all marginal values for these strata instead of sampling
  else
    for $j = 1 : m_k$ **do**
      Randomly generate a coalition S of size k
      Solve the Linear Program and save the basis of the LP problem
      for all player i not in the coalition S **do**
        Rebuild the model with new resource vector of $S \cup \{i\}$
        # Utilise LPSA to check if LP basis will changes;
        if the basic changes **then**
          Use dual simplex method to solve LP problem with rhs $\boldsymbol{b}(S \cup \{i\})$
        else
          Apply the closed form to calculate $\nu(S \cup \{i\})$
        end if
        Add $(\nu(S \cup \{i\}) - \nu(S))$ value into μ_i^k and σ_i^k calculations of player i
      end for
    end for
  end if
end for

formula $\phi_i = \frac{1}{n}\sum_{k=0}^{n-1} \mathbb{E}(X_i^k)$, the corresponding variances and expected values of stratified sampling for player i is calculated as following

$$\sigma_i^2 = \mathrm{Var}(\overline{\phi_i}) = \sum_{k=0}^{n-1} \mathrm{Var}(\frac{1}{n}\overline{X_i^k}) = \frac{1}{n^2}\sum_{k=0}^{n-1} \mathrm{Var}(\overline{X_i^k}) = \frac{1}{n^2}\sum_{k=0}^{n-1} (\sigma_i^k)^2$$

and

$$\mathbb{E}(\overline{\phi_i}) = \frac{1}{n}\sum_{k=0}^{n-1} \mathbb{E}(\overline{X_i^k}) = \frac{1}{n}\sum_{k=0}^{n-1} \mu_i^k = \phi_i$$

After the computation of the approximated Shapley value, we also implement the process of balancing the total budget (O'Brien et al. 2015) to guarantee the efficiency axiom. The process utilises the sampling averages and sampling variances of strata to find the maximum likelihood estimates of the Shapley value with budget constraints as follows:

$$\widehat{\phi}_i = \overline{\phi_i} - \frac{\sigma_i^2}{\sum_{j=1}^n \sigma_j^2}(\overline{\nu}(N) - \nu(N)),$$

where $\overline{\nu}(N) = \sum_{i=1}^{n} \overline{\phi_i}$. We note that $\mathbb{E}(\overline{\nu}(N)) = \sum_{i=1}^{n} \mathbb{E}(\overline{\phi_i}) = \sum_{i=1}^{n} \phi_i = \nu(N)$. Moreover, all four properties of Shapley value are satisfied in expectation for the LPSA Randomized algorithm as the result of the following equations:

$$\mathbb{E}(\widehat{\phi_i}) = \mathbb{E}(\overline{\phi_i}) - \frac{\sigma_i^2}{\sum_{j=1}^{n} \sigma_j^2}(\mathbb{E}(\overline{\nu}(N)) - \nu(N)) = \mathbb{E}(\overline{\phi_i}) = \phi_i,$$

and

$$\sum_{i=1}^{n} \widehat{\phi_i} = \sum_{i=1}^{n} \overline{\phi_i} - (\overline{\nu}(N) - \nu(N)) = \nu(N).$$

In the next section, we will show the effects of combining the LPSA for computing the marginal contribution of a player to a coalition and the proposed stratified sampling approach to evaluate the Shapley value of MIKG.

3.4 Computational Experiments

We first describe how to generate the data for some small and large-scale problems. The algorithms are implemented in MATLAB and run on an Intel-core i5-4570 PC with 3.2 GHz CPU and 4GB RAM. We first present results for small scale LPGs with $n \in \{10, 13, 16, 19, 22, 25\}$, for which the exact Shapley value can be computed using brute-force. Later, we extend the results to a set of larger size instances with up to 90 players and with only two types of resource vectors. This setting gives us the analytical form of the Shapley value, from which we can judge the accuracy of our approximation methods. Then, we show the effects of LP sensitivity analysis on the computational time of the LPSA randomised algorithm. Finally, we implement three other randomised algorithms in the literature to compare with the results.

For small-scale instances, we use the *mean absolute percentage error* (MAPE) as a measure between the approximate and the exact Shapley values, and which is calculated as: $\text{MAPE}(\boldsymbol{\phi}, \widehat{\boldsymbol{\phi}}) = 100.\frac{1}{n}\sum_{i=1}^{n} |\frac{\widehat{\phi_i} - \phi_i}{\phi_i}|$, where $\boldsymbol{\phi}$ is the real Shapley value and $\widehat{\boldsymbol{\phi}}$ is the approximate Shapley value.

3.4.1 Generating Problem Instances

Small scale LPG instances are generated as follows: For each tuple of n (the number of players), r (the number of resources) and p (the number of products), we generate a problem instance $(\mathcal{A}, \mathrm{B}, \boldsymbol{c})$ of the linear production game. The elements of $(\mathcal{A}, \mathrm{B}, \boldsymbol{c})$ are drawn randomly from a uniform distribution in the interval $[0, 1]$.

For large-scale instances, we consider a class of LPGs where there are only two types of players E and F, i.e., all players of the same type in the games have the same resource vector. The Shapley value can be determined exactly in closed form without finding 2^n values of $\nu(S)$. In particular, we generate a Linear production Game based on example 4.4 of Bjorndal & Jornsten (2009) with $p = 5$ products and $r = 10$ resources and $n = 40$ players. We describe the problem instances with two types of players E and F as follows:

$$\mathcal{A} = \begin{bmatrix} 7 & 3 & 5 & 2 & 1 \\ 6 & 9 & 9 & 5 & 10 \\ 6 & 3 & 3 & 4 & 3 \\ 9 & 5 & 4 & 2 & 1 \\ 3 & 6 & 10 & 2 & 4 \\ 4 & 5 & 1 & 3 & 8 \\ 4 & 3 & 4 & 2 & 3 \\ 7 & 9 & 1 & 1 & 7 \\ 5 & 8 & 9 & 3 & 2 \\ 2 & 6 & 3 & 10 & 2 \end{bmatrix}; \boldsymbol{c} = \begin{bmatrix} 53 \\ 57 \\ 49 \\ 34 \\ 41 \end{bmatrix}; \boldsymbol{b}_E = \begin{bmatrix} 4 \\ 0 \\ 9 \\ 0 \\ 19 \\ 0 \\ 17 \\ 4 \\ 28 \\ 0 \end{bmatrix}; \boldsymbol{b}_F = \begin{bmatrix} 15 \\ 40 \\ 11 \\ 22 \\ 7 \\ 22 \\ 0 \\ 22 \\ 0 \\ 23 \end{bmatrix}. \tag{3.9}$$

When we fix the total number n of players, suppose that the number of players of type E is $n_E = \lfloor 2/5 * n \rfloor$ and the number of players of type F is $n_F = n - n_E$. In the problem instance where $n = 40$, there are $n_E = 16$ players of type E and $n_F = 24$ players of type F, i.e. $(n_E = 16; n_F = 24)$. Note that all coalition S is composed of p_E players of type E and p_F players of type F, where $0 \leq p_E \leq n_E$ and $0 \leq p_F \leq n_F$. Therefore, the total number of coalition value $\nu(S)$ that we have to evaluate is $n_E.n_F$ instead of 2^n. This reduces the required computational effort for the experiment.

We evaluate the exact Shapley value of this game as $\phi_E = 191.03$ and $\phi_F = 57.88$. In this situation, the value of the grand coalition is $\nu(N) = 4445.6$. Although a coalition of only one type of players could not create any product, the balance between two types of players generates the most outcome. Player E has more payoff compared to player F because there are less number of type E in the game.

For each problem size (n, r, p), we generate some problem instances $(\mathcal{A}, \mathrm{B}, \boldsymbol{c})$ to test our approximation algorithm. With each problem instance, the approximation algorithm output different results for each runs (trials). Hence we take the average value for these calculations in the final result.

3.4.2 Effects of using LP Sensitivity Analysis

Consider an LPG where $p = 19$ products and $r = 27$ resources, we change the number of players to investigate the LP sensitivity analysis. Suppose we have a total of 10000 samples to distribute in all the computation of the LPSA randomised algorithm. Let us denote P_{bnc} as the percentage of times in the process of calculating marginal values in which the calculation of the marginal contribution $\nu(S \cup \{i\}) - \nu(S)$ does not need to change the basis (i.e. can be calculated in closed form). Let T_{SA} and T_{woSA} be the calculation times (in seconds) for the algorithms with and without applying the LP sensitivity analysis. We also show the exact time to compute the Shapley value with label T_{exact} for comparison.

Table 3.1 shows these statistics over 100 trials for this problem size with a different number of players. Note that as the number of players in the game is small, it is feasible to compute the exact Shapley value without randomised methods, for example, the number of coalition values is $2^n < 10^4$ when $n \leq 13$. However, as the number of players increases, computational time of finding the correct Shapley value rise exponentially. In particular, the computational time of the exact Shapley value is more than ten hours for $n \geq 25$ as shown as $*$ in the table.

n	13	16	19	22	25	28
$P_{bnc}(\%)$	7.68	12.37	15.9	14.16	15.83	16.39
$T_{SA}(s)$	41.02	37.8	35.84	35.82	34.75	34.42
$T_{woSA}(s)$	42.43	41.16	40.48	39.83	39.36	38.71
$T_{exact}(s)$	21.63	150.81	1415	9514	*	*

Table 3.1: Average 100 trials for problem instances of small sizes with $p = 19, r = 27$ and total 10000 samples.

For larger instances, we generate ten problem instances for each choice of n and implement 10 trials for each problem instance with fixed number of players. Table 3.2 shows similar computational results in those large LPGs without the exact Shapley value time. Tables 3.1 and 3.2 show the effect of using sensitivity analysis in our randomised algorithms for calculating the approximate Shapley value.

When the number of players n increase, the use of sensitivity analysis is more effective as each player can contribute proportionally more to a small group compared to a bigger one; hence it is worth to use sensitivity analysis in the case of large games. For example, in Table 3.2, when the number of players n increases to greater than 30, the P_{bnc} shows that there are more than 20% of the total 10000 samples in which we could save time in the marginal contribution calculations. Therefore, the computational time of the

n	30	40	50	60	70	80	90
$P_{bnc}(\%)$	19.23	27.48	32.67	35.93	35.68	37.12	45.16
$T_{SA}(s)$	34.11	28.82	26.51	26.51	26.27	24.59	21.36
$T_{woSA}(s)$	38.64	37.86	37.02	35.87	38.43	35.25	35.87

Table 3.2: Average 10 trials for 10 problem instances of large size with $p = 19, r = 27$ and total 10000 samples.

algorithm with LPSA is significantly less than the method without LPSA when the number of players increases.

3.4.3 Effects of using the Stratified Sampling technique

In this part, we implemented and compare three other approximation algorithms: Simple random sampling (adapted from Castro et al. 2009), Stratified coalition sampling (adapted from Maleki et al. 2013), and Learning stratified sampling (adapted from O'Brien et al. 2015) with our LPSA randomised algorithm. We apply the Linear Programming Sensitivity Analysis technique for all calculation of marginal contributions in these algorithms. More detail of the different steps among these sampling techniques can be found below:

Simple random sampling

With the Shapley formula (3.3) of permutations, we can apply directly the simple random technique to approximate the Shapley value. This is simple to implement compared to other randomised methods because of its formulation. The method has some advantages such as being unbiased and having no error for dummy players. Moreover, its theoretical error of the approximation can be calculated in a probabilistic way.

Stratified coalition sampling

This sampling approach utilises the coalitional formula (3.2) of the Shapley value to design the stratified sampling technique instead of computing all 2^n coalitions. Each stratum k will be a set of all coalitions of the same size k. The authors propose an heuristic method to allocate the samples into different strata. The approach provides a non-asymptotic bound for the sampling error by using Hoeffding's inequality (Hoeffding 1963, Theorem 2.) to estimate the error within each stratum.

Learning stratified sampling

In this stratified algorithm, the Neyman allocation is applied to decide the numbers of

sampling units taken for each stratum. A reinforcement learning algorithm is proposed in the paper by O'Brien et al. (2015) to estimate the strata' variances and adjust the sample allocation among strata. We also need to identify the proportion of 'exploration stage' and 'exploitation stage' to determine the sampling allocation, which is a heuristic step in practice. In this implementation of the learning algorithm, we use the same coefficients γ and β of the previous paper for our sigmoid function.

We will create some randomly generated problem instances $(\mathcal{A}, B, c)$ with the parameters $p = 19$, $r = 27$ and $n = 22$. The Shapley value is computed by four different algorithms with their MAPE values to compare the sampling errors. Figure 3.2 plots MAPEs against the time budgets to compare convergence speeds of the algorithms. Note that; we implement the algorithm with a different number of samples for each player. The total number of samples to take for each sampling algorithm depends on the running time of that algorithm; hence there is the trade-off between the exactness and computational time.

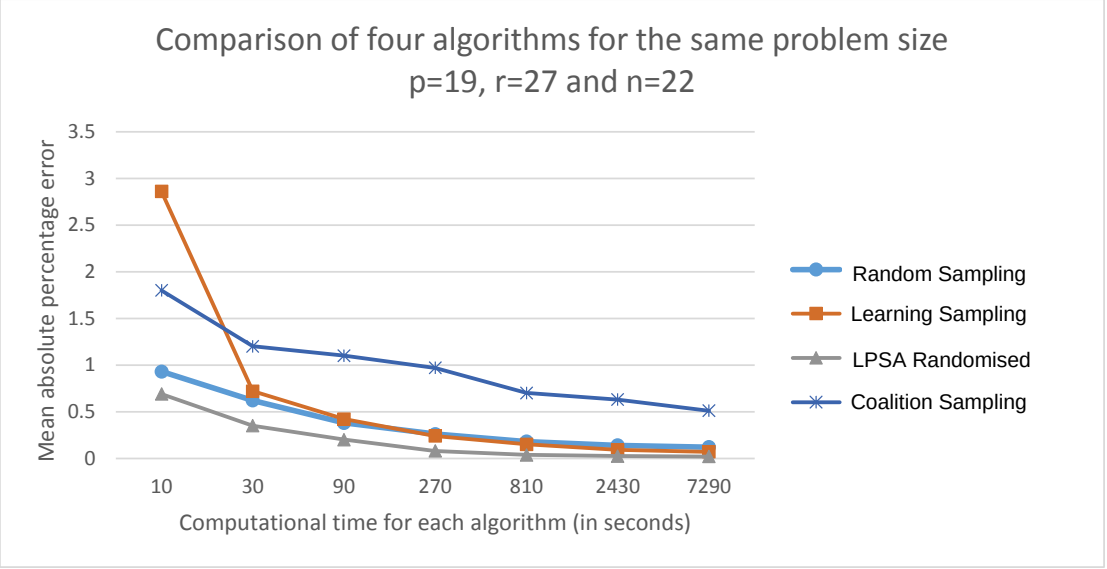

Figure 3.2: Comparison of four algorithms using different sampling techniques

In Figure 3.2, the MAPEs of all algorithms decrease to less than 1% as each algorithm runs more than five minutes, i.e. the number of samples also increases proportionally. However, there are some differences in computational times among these algorithms, the coalition sampling is not competitive compared to three other algorithms. The LPSA randomised algorithm is the best compared to others w.r.t. MAPE, especially when there is less computational time.

In Figure. 3.3, we include a boxplot which presents the spread of sampling errors for the four different algorithms with the same running time for the large-scale problem instances

with $p = 5, r = 10$ and $n = 40$ in equation (3.9). These results show that, with the same budget of computational time, the LPSA randomised algorithm is the best sampling algorithm on average compared to other sampling approaches (w.r.t. MAPE) for finding the Shapley value. The coalition sampling and the Learning stratified sampling are not effective in this problem instances compared to the simple random sampling and LPSA randomised algorithms.

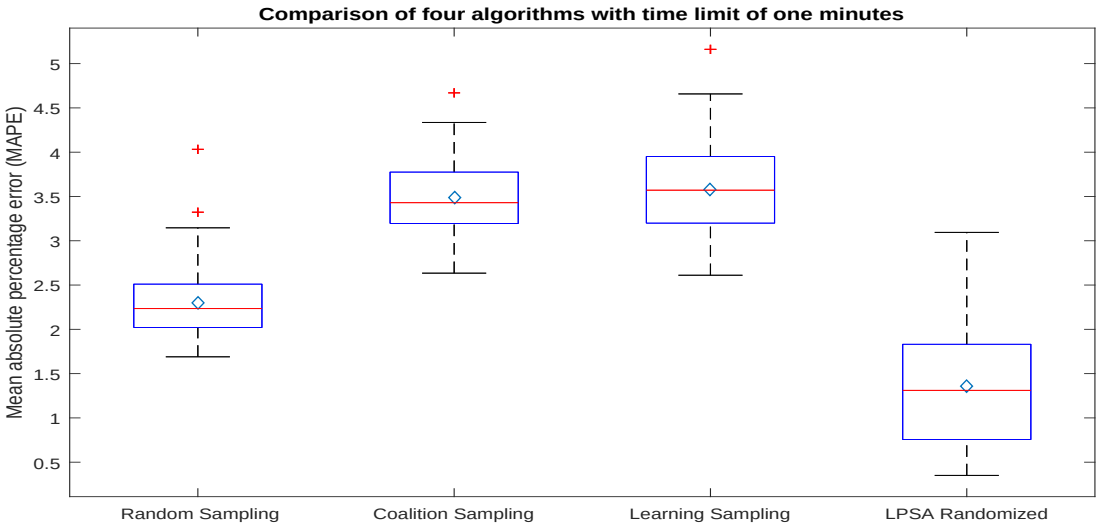

Figure 3.3: The MAPE box-plot of four algorithms to approximate the Shapley value of a large-scale LPG

The table lists the changes of the MAPE for four algorithms with different values of number n of players. With each value n, we generate ten problem instances and calculate the ten trial average of those MAPEs of each algorithm. It shows that if we set the time fixed, the LPSA randomised algorithm is on average the best compared to other algorithms in the literature.

Number of players	30	40	50	60	70	80	90
Random sampling algo.	2.48	3.81	3.77	4.51	4.78	3.94	5.03
Coalition sampling algo.	3.93	5.44	5.78	6.50	6.73	6.14	6.63
Learning algo.	4.26	5.88	5.10	5.70	5.53	7.18	6.98
LPSA Randomized algo.	1.56	1.37	2.11	2.39	2.52	2.93	3.65

Table 3.3: Average MAPE of 10 trials for 10 problem instances of size $p = 5, r = 10$ with five minute time limit.

3.4.4 Error Estimation for the LPSA Randomized algorithm

We generate some numerical results presenting the combined effects of both LPSA and the stratified sampling method, and show that they work well with each other. The test instances include some large-scale games with two types of players as described in the Section 3.4.1.

Initially, we present some experiments with the LPSA Randomized algorithm if the sampling budget and number of players in the game are modified. When the total number of players n ranges from 30 to 90, the total sampling budget m increases from 1000 to $3^5.1000$. We then describe the MAPEs of the algorithm showing the computational errors between the real Shapley values and the calculated values in Table 3.4. In general, a game with a fixed number of player has smaller error if having more samples to approximate the Shapley value. As the number of players increase, the table indicate that we have to raise the sampling budget to keep a small sampling error.

$m \setminus n$	30	40	50	60	70	80	90
1000	10.48	13.76	15.36	14.48	11.96	13.74	16.39
3000	7.59	8.78	8.20	10.32	8.76	12.22	11.12
9000	4.63	4.09	5.52	4.71	6.75	6.18	6.29
27000	2.18	2.52	2.62	2.33	3.80	4.57	3.87
81000	1.66	1.26	1.75	1.41	1.66	1.58	1.47
243000	0.69	0.68	0.87	0.93	0.85	1.15	0.97

Table 3.4: MAPEs of LPSA Randomized algorithm for large-scale problem instances with different sample sizes

Afterwards, some test instances of $p = 5, r = 10$ and $n = 40$ will be shown in Table 3.5, where there are different numbers of players for two types E and F. For these instances, we also use the *root mean squared error* (RMSE) as a quadratic scoring rule which measures the average magnitude of the error. The RMSE is calculated as: $\text{RMSE}(\boldsymbol{\phi}, \widehat{\boldsymbol{\phi}}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\widehat{\phi}_i - \phi_i)^2}$, where $\boldsymbol{\phi}$ is the real Shapley value and $\widehat{\boldsymbol{\phi}}$ is the approximate Shapley value. The real Shapley values and the average MAPEs and RMSEs of four algorithms are described as below:

In Table 3.5, the less the number of players of one type increases the percentage of values that player type can claim from the $\nu(N)$ in the Shapley value. Moreover, the grand coalition value is largest when there is a balance between two types of players. The reason that some problem sizes have large MAPEs is the Shapley values of one player become very small, and any slight changes in the value of that approximate Shapley

# players of (E, F)	(5,35)	(10,30)	(15,25)	(20,20)	(25,15)	(30,10)	(35,5)
Shapley value of E	279.3	256.6	211.8	95.5	18.5	4.7	1.2
Shapley value of F	1.9	11.3	41.8	145.5	215.1	232.4	238.2
$\nu(N)$	1463	2905	4221	4820	3689	2465	1233
MAPEs	14.56	7.42	2.1	0.64	4.55	16.98	28.58
RMSEs	0.87	1.85	1.54	0.81	2.31	1.84	0.95

Table 3.5: The approximation errors of LPSA Randomized algorithm on large-scale problem instances with five minutes time limit.

value will affect the MAPEs of the algorithm. The RMSEs confirm this argument as their errors are quite small w.r.t. the time limit.

3.5 Conclusions

In this chapter, we describe new computational methods for calculating the Shapley value of linear production games. We show how to find the closed form solution of the Shapley value in some particular cases. For large general LPGs, we proposed a randomised algorithm combining both Linear Programming sensitivity analysis and stratified sampling method. The computational results compare the LPSA randomised methods with some other sampling techniques in the literature and show that our method clearly outperforms others. We expect that these LPSA randomised algorithm could be used to approximate the Shapley value for some other cooperative games such as minimum spanning tree games, assignment games and network synthesis games.

Chapter 4

The Shapley value of Integer Optimisation Games

4.1 Introduction

This chapter focuses on a class of cooperative games called Integer Optimisation game for which the characteristic function arises from an integer programming problem. A closely defined subclass of Operational Research game is the Combinatorial Optimisation game, where the payoff of each coalition S is defined as the objective outcome of a combinatorial optimisation when the subset S of players decides the underlying combinatorial structure. In this direction of development, we now present some of the general subclasses of Operational Research games including the packing games, combinatorial optimisation games, and integer optimisation games. These generalised games give us a broad sense of the connection between mathematical programming and cooperative game theory.

As we mentioned in the previous chapter of linear production game, the model of Owen (1975) shows a group of resource owners collaborate to produce goods which can be sold at a given market price. As the collaboration outcome of each group is the optimal value of a linear programming problem, the players need a good strategy to share the payoff among themselves. The Owen's model is useful but many operations research games in practice can not be formulated as linear production game. Therefore, a natural extension of this model is in the paper of Granot (1986), where the combined resource function does not have the assumption of additivity property similar to the linear production game. The authors also formulated several cooperative games which cannot be handled by Owen's model such as minimum spanning tree game, weighted matching game and network synthesis game.

Another subclass of cooperative games is combinatorial optimisation games, in which the value of the coalition S can be computed by solving a combinatorial optimisation problem. The combinatorial optimisation games cover some other popular games such as the minimum cost spanning tree games (Granot & Huberman 1981), the assignment game (Shapley & Shubik 1971), the facility location game (Goemans & Skutella 2004), the travelling salesman games (Tamir 1988), and the vehicle routing game (Göthe-Lundgren et al. 1996). The covering/packing game is another combinatorial optimisation game, which is described in Deng et al. (1999). The authors describe this class of cooperative games with the characteristic function value defined by an integer program:

$$\nu(S) := \max \ \{\boldsymbol{c}^T \boldsymbol{x} \ \mid \ \mathcal{A}\boldsymbol{x} \leq \boldsymbol{b}, \ \boldsymbol{x} \in \{0,1\}^q\}, \tag{4.1}$$

where $\mathcal{A}$ is a binary matrix of size (r, q) and $\boldsymbol{b}$ is a vector with binary entries. This subclass of the combinatorial optimisation game includes many classic type of games such as the covering games, max-flow game, packing games and minimal colouring games.

More recent generalization of the combinatorial optimisation games is defined as Integer Minimisation game in Caprara & Letchford (2010). The paper defined the total cost of each group coalition as the optimal result of an Integer Linear Program:

$$c(S) := \min\{\boldsymbol{c}^T \boldsymbol{x} \ \ : \ \ \mathcal{A}\boldsymbol{x} \geq \mathrm{B}\,\boldsymbol{y}(S) + \boldsymbol{d}, \ \boldsymbol{x} \in \mathbb{Z}_+^q\}, \tag{4.2}$$

where $\boldsymbol{y}(S)$ is the incidence vector of subset $S \subset N$, $\boldsymbol{d} \in \mathbb{Z}^r$ and $\mathcal{A}, \mathrm{B}$ are integer matrices of dimension (r, q) and (r, n). Note that all combinatorial games can be formulated as an integer minimisation game by introducing maximum one variable for each coalition. The authors also considered the facility location, travelling salesman and vehicle routing games, which have been formulated as homogeneous Integer Optimisation Games, and are therefore super-additive.

The similar version of the Integer Minimisation Game, where the cooperative games are in the value form, is Integer Maximisation Game. Using the same notation of $(\mathcal{A}, \mathrm{B}, \boldsymbol{c}, \boldsymbol{d}, \boldsymbol{y}(S))$, the value function of a coalition S has the form:

$$\nu(S) := \max\{\boldsymbol{c}^T \boldsymbol{x} \ \ : \ \ \mathcal{A}\boldsymbol{x} \leq \mathrm{B}\boldsymbol{y}(S) - \boldsymbol{d}, \boldsymbol{x} \in \mathbb{Z}_+^q\}. \tag{4.3}$$

The super-additive property of the integer maximisation games can be deduced when the condition of $\boldsymbol{d} \geq 0$ is satisfied, i.e., for any two disjoint coalitions S and S', we have $\nu(S \cup S') \geq \nu(S) + \nu(S')$. In this case, it is always beneficial for the players to work together in a grand coalition. The same property can be proved for the integer minimisation games. To keep it simple, we call the class of cooperative games contains both of these games as the *Integer Optimisation Game* (IOG).

In this chapter, we introduce a new class of cooperative games called *Multidimensional Integer Knapsack Game* (MIKG). This class is a type of the homogeneous integer maximisation games, hence, is a super-additive game. The MIKG is the cooperative game version of the *Multidimensional Integer Knapsack Problem* (MIKP), which has attained a lot of attention in the literature. Although the classical knapsack problem can be solved in pseudo-polynomial time using dynamic programming, computing the multidimensional integer knapsack problem is a NP-hard problem (Bertsimas & Demir 2002, Puchinger et al. 2010). We later show the connection of the MIKG model with some other cooperative games in the literature and several practical applications of the MIKG are presented.

Among many solution concepts for MIKG, we select the Shapley value to allocate the payoff because of its 'fair' properties: efficiency, dummy, symmetry, additivity, and monotonicity. Although the Shapley value can be defined for non-supperadditive cooperative game (Aumann 2009), where the coalition structure is not a grand coalition, we will only focus on the case of computing the Shapley value for supper-additive IOG. The *Multidimensional Integer Knapsack Problem* (MIKP) has attained a lot of attention in the literature. Although the classical knapsack problem can be solved in pseudo-polynomial time using dynamic programming, computing the multidimensional integer knapsack problem is a NP-hard problem (Bertsimas & Demir 2002, Puchinger et al. 2010). Therefore, solving the multidimensional knapsack problems 2^n times for the MIKG characteristic function to find the Shapley value could be time-consuming.

In general, computing the Shapley value for Integer Optimisation Game is a challenging task due to its combinatorial structure. Therefore, we propose an algebraic approach of Gröbner bases to find the Shapley value of the IOG with the super-additive property, in particular, this chapter focuses on the MIKG model. In the general case, we can modify our algorithm for MIKG by adding/reducing the vector $\boldsymbol{d}$ to the right hand side of the integer program formula. In this chapter, the main contributions of our work include:

- We introduce a new class of cooperative games called the Multidimensional Integer Knapsack Games. The connection between MIKG and other games in the literature and some practical applications are given.
- The algebraic approach of Gröbner bases is utilised in solving the integer optimisation problem and then to find the exact Shapley value of MIKG for small scale problems.
- For the class of MIKG with a large number of players ($20 \leq n \leq 100$), we combined the algebraic Gröbner approach with the sampling techniques to approximate the Shapley value.

- Finally, the computational experiments show the effectiveness of the proposed methods compared to CPLEX solver.

The chapter is organised as follows: In Section 4.2 the preliminaries give some background about the Gröbner method to solve Integer Program. The Section 4.3 formalises the formulation of the MIKG, its applications and presents some properties of its Shapley value. In Section 4.4, we use an augmentation algorithm with the Gröbner basis to propose a framework to find the exact Shapley value for small-scale games and approximate Shapley value for large scale games. Section 4.5 contains some computational experiment and shows the numerical results for different problem instances.

4.2 Preliminaries

This part introduces the Algebraic Gröbener method in computer algebra and its application to find out the test set for Integer Programs. We consider an integer linear programming problem $ILP(\boldsymbol{b})$ in augmented form with right hand side (rhs) vector $\boldsymbol{b}$ of dimension r.

$$\begin{aligned} \mathrm{ILP}_{(\mathcal{A},\boldsymbol{c})}(\boldsymbol{b}) := \quad \max \quad & \boldsymbol{c}^T\boldsymbol{x} & \\ \text{s.t.} \quad & \mathcal{A}\boldsymbol{x} & = \boldsymbol{b}, \\ & \boldsymbol{x} & \in \mathbb{Z}^m_+, \end{aligned} \tag{4.4}$$

where $\mathcal{A} \in \mathbb{Z}^{r\times m}$, $\boldsymbol{b} \in \mathbb{Z}^r$, $\boldsymbol{c} \in \mathbb{R}^m$. The notation $\mathrm{ILP}_{(\mathcal{A},\boldsymbol{c})}(\boldsymbol{b})$ denotes the integer linear programming problem with a fixed matrix $\mathcal{A}$, a cost vector $\boldsymbol{c}$, and a right-hand-side $\boldsymbol{b}$. Let us call the family of integer programs which fixed $(\mathcal{A}, \boldsymbol{c})$ and varied $\boldsymbol{b}$ as $\mathrm{ILP}_{(\mathcal{A},\boldsymbol{c})}$.

Several techniques for solving these integer linear program problems exist (see, for example, the book by Wolsey & Nemhauser 1999). In this part, we investigate one such solution method using test set for integer programming. A test set of vectors is a finite set of vectors such that it allow to improve any given feasible non-optimal point of an integer program by moving along at least one vector in the set. If this test set exists, the program can be solved to optimality by the augmentation algorithm if we are given an initial feasible solution. There are three test sets of ILP can be found in the literature including the Scarf test set(neighbors of the origin)(Scarf 1981), the Gröbner basis (Conti & Traverso 1991), and the Graver basis (Graver 1975). The paper of Weismantel (1998) discusses the relationship among these test sets and the connection between the test set and the augmentation problem for integer programming.

For the MIKG, we focus on the Gröbner basis approach because of its advantages on the size and computational time of the test set. This algebraic Gröbner approach was recently known as a new approach for solving integer programs (see for examples, Thomas 1995). In the paper of Tayur et al. (1995), the authors exploit the structure of these integer programs and certain features of the algorithm to compute the test sets and the exact optimal solutions for a family of constrained integer programs that arose in a manufacturing setting. Bertsimas (2000) shows the Gröbner method presenting a generalization of the Farkas lemma for ILP, hence, provides a way to check whether a given problem is infeasible. The method also gives the structural information of the feasible set and helps performing sensitivity analysis for ILP.

The Gröbner basis theory is a well-established research area in commutative algebra and algebraic geometry. The Gröbner basis for a given polynomial ideal with respect to some fixed term order was introduced in 1965 in Bruno Buchberger's PhD dissertation. Gröbner basis is a special generator of polynomial ideals which can be computed using Buchberger's algorithm (Buchberger 1965). The algorithm can be considered as the generalization of the Euclidean algorithm for computing the univariate greatest common divisor and of Gaussian elimination for linear systems.

The first paper that describes an algebraic method to solve integer programs of the standard form, using ideas from Gröbner basis theory, is by Conti & Traverso (1991). For the family of $\text{ILP}_{(\mathcal{A},c)}$, a geometric interpretation of the Conti-Traverso algorithm provides a unique minimal test set, which is called the reduced Gröbner basis. In practice, these test sets of Gröbner bases of polynomial ideals can be computed using a computer algebra package likes 4ti2 (4ti2 2016). From now on, we call this method for solving Integer Programming as the *Algebraic Gröbner*(AG) method. More detail on the relationship between the Gröbner bases of algebraic geometry and the test set theory in integer programming can be found in appendix A.3.

The computation of these Gröbner bases depends significantly on the Buchberger's algorithm. In the literature, there are a few methods to implement the Buchberger's algorithm (for example, see the paper of Hemmecke & Malkin (2009)). This paper shows that the Project and Lift algorithm is the best method compared to others such as BLR algorithm (Bigatti et al. 1999) and Saturation algorithm (Hosten & Sturmfels 1995). In general, this method can be applied to solve the system of equations, hence their complexity is at least that of solving polynomial system. As a result, computing the Gröbner bases is an NP-hard problem (for more detail, see Cox et al. 1998).

In the literature, there are a few applications of the AG methods to linear (and non-linear) Integer Programs. Hemmecke et al. (2011) show some linear integer program applications in practice such as the congestion-avoiding routing and the error-correcting

codes problems. Two other non-linear examples are the cost minimisation of scheduling jobs on parallel machines given restrictions on demands and capacity (Gago et al. 2013) and the cost minimisation problem in the series parallel redundancy allocation problem given a target reliability (Gago et al. 2015). This algebraic approach is also implemented for the portfolio optimisation problem by solving a dual search strategy using the Gröbner basis considered as test sets (Castro et al. 2011). In some special classes such as the transportation problems and network flow problems, the Gröbner bases can be calculated in polynomial time w.r.t the input. More detail about the theory of using the algebraic and geometric method in discrete optimisation can be found in the book by Loera et al. (2013).

4.3 Multidimensional Integer Knapsack Games (MIKG)

We now introduce the Multidimensional Integer Knapsack Game as a typical example of the homogeneous class of Integer Maximisation Game. It can also be derived from the class of linear production games when we restrict the decision variables to be the integers. There is a strong connection of this class of cooperative game with the skill games in the literature. Many practical versions of MIKG inspired from the applications of the MIKP such as the bin packing, cargo loading, cutting stock problems, processors allocation in huge distributed system, and capital budgeting.

4.3.1 Definition

A multi-dimensional knapsack game (N, ν) is a cooperative game with the set of players $N = \{1, \ldots, n\}$ and each player i has a vector of resource capacities $\{b_i^k\}_{k=1}^r$. Each coalition of players S can choose a set of items in $J = \{1, \ldots, q\}$ and there is no upper bound on the number of copies for each kind of item. Let denote $\boldsymbol{p} = \{p_j\}_{j=1}^q$, where p_j is the value of the item j, and the weight of that knapsack item j is given by a r-dimensional integer vector $\boldsymbol{w}_j = (w_{1j}, \ldots, w_{rj})$. The players in the coalition aim to maximise the total value, however, they have to satisfy all knapsack constraints. The MIKG characteristic function can be defined by the following Integer Linear Program:

$$\begin{aligned} \nu(S) := \quad & \max \sum_{j=1}^{q} p_j z_j \\ \text{s.t.} \quad & \sum_{j=1}^{q} w_{kj} z_j \leq b^k(S); \quad k = 1, \ldots, r, \\ & z_j \in \mathbb{Z}_+ \qquad\qquad\quad j = 1, \ldots, q. \end{aligned} \tag{4.5}$$

where $b^k(S) := \sum_{i \in S} b_i^k$ is the amount of resource $k = \{1, \ldots, r\}$ of coalition S.

From the definition of the MIKG, we can transform the characteristic function into standard form with new slack binary variable vector $\boldsymbol{y}$. Therefore the problem (4.5) can be rewritten as:

$$\nu(S) := \max \quad \{\boldsymbol{p}^T \boldsymbol{z} : \quad \mathrm{W}\boldsymbol{z} + \mathrm{I}_r \boldsymbol{y} = \boldsymbol{b}(S), \quad \boldsymbol{z} \in \mathbb{Z}_+^q, \boldsymbol{y} \in \mathbb{Z}_+^r\}, \tag{4.6}$$

or equivalently,

$$\nu(S) := \max \quad \{\boldsymbol{c}^T \boldsymbol{x} : \quad \mathcal{A}\boldsymbol{x} = \boldsymbol{b}(S), \quad \boldsymbol{x} \in \mathbb{Z}_+^m\}, \tag{4.7}$$

where the matrix $\mathcal{A} = [\mathrm{W}|\mathrm{I}_r]$, and the vectors $\boldsymbol{x} = [\boldsymbol{z}, \boldsymbol{y}]$ and $\boldsymbol{c} = [\boldsymbol{p}, 0_r]$ have sizes $m = q + r$. The 0_r is the zeros vector of length r and I_r is the identity matrix of size (r, r). Let us also denote $\mathrm{B} := \{b_i^k\}_{(r \times n)}$ as the resource matrix of the players in this game. In the formulation (4.7), the resource vector $\boldsymbol{b}(S) := \{\boldsymbol{b}^k(S)\}_{k=1}^r = \mathrm{B}\,\boldsymbol{y}(S)$, where $\boldsymbol{y}(S)$ is the incidence vector of the subset S.

We first show that the MIKG is a super-additive game, where the players have the incentive to build a grand coalition for their benefits. For two coalition S and S' of players, if let denote the two solution of the Integer Programs are $\boldsymbol{x}(S)$ and $\boldsymbol{x}(S')$. Then their sum $\boldsymbol{x}(S) + \boldsymbol{x}(S')$ is also a feasible solution for the Integer Program of the coalition $S \cup S'$. Therefore, the optimal value of the union coalition must satisfy the condition: $\nu(S \cup S') \geq \nu(S) + \nu(S')$. This property helps us solve the coalitional generation structure, and brings the focus to the payoff allocation problem.

4.3.2 Applications of MIKG and related classes of cooperative games

This section aims to propose some applications for the multidimensional integer knapsack game and to connect it with some other cooperative games in the literature. The general multidimensional integer knapsack problem and its variants have been widely used to model many practical problems such as project selection, capital budgeting, combinatorial auctions, and inventory allocation in assemble-to-order systems, among others. We now present some similar applications for the MIKG as follows:

- The integer production games, where the underlying optimisation problem is similar to the Linear production games (Owen 1975) with integer decision variables, is one typical example. When each player has a different resource vector and the outcomes of the production process must have the integer value (i.e. there are

integer amounts of products of each type to produce), our MIKG model is the generalised version of the LPG.

- In capital budgeting and project selection applications (Weingartner 1966, Peterson 1967) the MIKG appears when there are a set of investors with different resource vectors who want to collaborate for investment in some time periods. This problem is a different version of the MIKG with the solutions are the binary vector, and the constraints reflect maximum permissible expenditure or the incremental cash balance in each time period. The value of the characteristic function $\nu(S)$ is then the optimal outcome of a binary integer program.

- Another example of the MIKG is the cutting stock games in the paper industry. Suppose that the paper rolls have fixed width, some customers who want to collaborate and order for rolls of smaller widths. The constraints show that the total rolls of a fixed width produced must be greater than the total demand of that roll size. The cutting stock problem (Gilmore & Gomory 1963) aim to determine how to cut the rolls into smaller widths to satisfy the orders such that it can minimise the amount of scrap. For the cutting stock games, we also need to identify the payments of each customer if they work together in a coalition.

Another version of the MIKG called the Knapsack Budgeted Games, which has been investigated in Bhagat et al. (2014). Compared to our proposed games, this type of game is a binary knapsack game and the value of a coalition S of players is determined by a critical subset $T \subset S$ due to a physical or budget constraint which restricts the size of T. The Knapsack Budgeted Games usually can be found in sports, where at most k players from each team can play in a match. The value of a team is the total aggregate skill levels of its best k players since they play in most part of the games.

The MIKG is also partially related to the Coalitional Resource Games (CRG) in the cooperative game theory literature (Wooldridge & Dunne 2006). In both classes of games, the players have different kinds of 'resources' with the resources vector in MIKG and the endowment function in CRG. For the technology process, the MIKG has the weight of knapsack items and the CRG has a requirement function, which shows the quantity of each resources required to achieve a goal, to model the games. With respect to the outputs, the coalitions of agents collaborate to build finished goods in MIKG and complete goals in CRG. However, the two games have some major distinctions because of their different modelling purposes.

The main difference for the MIKG and the CRG is that the former is a Transferable Utility game, but the latter is a Non-Transferable Utility one. Another difference between the two models is the ways the two game-models present their goals' success.

The MIKG model maximises the total profit for selling the final goods in the grand coalition, but the CRG has to decide which coalition each player will join in the final coalition structure. Most of the previous researches about coalitional resource games utilised the core in the stable coalition structure instead of the Shapley value for the solution concepts as in this chapter.

4.3.3 Analytical properties of the Shapley value of the MIKG

The Shapley value of a player i is defined as the weighted average of all marginal contribution $\nu(S \cup \{i\}) - \nu(S))$ of that player for all coalition $S \subseteq N \setminus \{i\}$. Therefore, in the MIKG model, we are interested in how the Shapley value ϕ_i of a player i changes if the cost vector $\boldsymbol{c}$ changes to $\boldsymbol{c}'$ in the formula (4.7) of $\nu(S)$. We show that instead of recalculating all coalitional values $\nu(S)$ in this situation, the Shapley value can be quickly computed when the vector $\boldsymbol{c}'$ is quite close to the vector $\boldsymbol{c}$.

In the paper of Sturmfels & Thomas (1997), the authors investigate the polyhedral results that follow the Gröbner basis approach for solving Integer program when the cost vector of the objective function changed. Suppose that the feasible polytope of problem $ILP_{(\mathcal{A},\boldsymbol{c})}(\boldsymbol{b})$ is a bounded polyhedron. Since the $(r \times m)$-matrix $\mathcal{A} = (a_1, \ldots, a_m)$ has rank r, let define a lattice $ker_{\mathbb{Z}}(\mathcal{A}) := ker(\mathcal{A}) \cap \mathbb{Z}^m$ and a semigroup $cone_{\mathbb{N}}(\mathcal{A}) := \{\sum_{i=1}^{m} \lambda_I a_i : \lambda_i \in \mathbb{Z}_+\}$. Then the integer program $ILP_{(\mathcal{A},\boldsymbol{c})}(\boldsymbol{b})$ is feasible if an only if $\boldsymbol{b}$ lies in the semigroup $cone_{\mathbb{N}}(\mathcal{A})$.

Definition 4.1. Two cost vectors $\boldsymbol{c}$ and $\boldsymbol{c}'$ in $\mathbb{R}^m$ are equivalent (w.r.t. $\mathcal{A}$) if the integer programs $ILP_{(\mathcal{A},\boldsymbol{c})}(\boldsymbol{b})$ and $ILP_{(\mathcal{A},\boldsymbol{c}')}(\boldsymbol{b})$ have the same set of optimal solutions for all $\boldsymbol{b}$ in $cone_{\mathbb{N}}(\mathcal{A})$.

Let π be the map $\pi(\boldsymbol{x}) = \mathcal{A}\boldsymbol{x}$ for $\boldsymbol{x} \in \mathbb{Z}_+^n$, the set $\pi^{-1}(\boldsymbol{b}) = \{u \in \mathbb{Z}_+^n : \pi(u) = \boldsymbol{b}\}$ is called the $\boldsymbol{b}$-fibre of $ILP_{(\mathcal{A},\boldsymbol{c})}$ for each a vector $\boldsymbol{b} \in \mathbb{Z}^d$. The cost vector $\boldsymbol{c}$ is assumed to be generic for $ILP_{(\mathcal{A})}$, i.e. the optimal solution with respect to $\boldsymbol{c}$ in every $\boldsymbol{b}$-fibre of $ILP_{(\mathcal{A},\boldsymbol{c})}$.

Proposition 4.2. *(Prop. 3.2. in Sturmfels & Thomas (1997)) Given two generic cost function $\boldsymbol{c}$ and $\boldsymbol{c}'$ in $\mathbb{R}^m$, the following are equivalent:*

- *For every $\boldsymbol{b} \in cone_{\mathbb{N}}(\mathcal{A})$, the programs $ILP_{(\mathcal{A},\boldsymbol{c})}(\boldsymbol{b})$ and $ILP_{(\mathcal{A},\boldsymbol{c}')}(\boldsymbol{b})$ have the same optimal solution.*
- *The cost functions $\boldsymbol{c}$ and $\boldsymbol{c}'$ support the same optimal vertex in each fiber $\pi^{-1}(\boldsymbol{b})$ of $ILP_{(\mathcal{A},\boldsymbol{c})}$.*

- *The reduced Ggröbner bases* $\mathcal{G}_{\mathcal{A},\prec_c}$ *and* $\mathcal{G}_{\mathcal{A},\prec_{c'}}$ *are equal.*

Using this equivalent relationship, the authors construct a polytope $St(\mathcal{A})$ whose normal cones are these equivalent classes. The cones are described by the inequalities in the reduced Gröbner bases associated with $\mathcal{A}$. When the cost vector c varies, they prove that the union of the reduced Gröbner bases $\mathcal{G}_{\boldsymbol{c}}$ consists precisely of the edge directions of $St(\mathcal{A})$ and also present the geometric algorithms for computing the universal Gröbner basis. We can extend this result to the problem of calculating Shapley value as follows.

Theorem 4.3. *For the previous MIKG model, if two cost vectors* c *and* c' *are equivalent, then the Shapley value of each player* i *changes proportionally w.r.t. these cost vectors.*

Proof. We now consider the Shapley value $\phi_{i,\boldsymbol{c}}$ of the player i w.r.t. cost vector $\boldsymbol{c}$ of the MIKG as follows:

$$\begin{aligned}
\phi_{i,\boldsymbol{c}} &= \sum_{S\subseteq N\setminus\{i\}} \frac{|S|!(n-|S|-1)!}{n!} && [\nu(S\cup\{i\})-\nu(S)] \\
&= \sum_{S\subseteq N\setminus\{i\}} \frac{|S|!(n-|S|-1)!}{n!} && [\boldsymbol{c}^T\{\boldsymbol{x}_{\boldsymbol{c}}(S\cup\{i\})\}-\boldsymbol{c}^T\{\boldsymbol{x}_{\boldsymbol{c}}(S)\}] \\
&= \boldsymbol{c}^T \sum_{S\subseteq N\setminus\{i\}} \frac{|S|!(n-|S|-1)!}{n!} && [\boldsymbol{x}_{\boldsymbol{c}}(S\cup\{i\})-\boldsymbol{x}_{\boldsymbol{c}}(S)] \\
&= \boldsymbol{c}^T[x^i_{1,\boldsymbol{c}},\ldots,x^i_{n,\boldsymbol{c}}]
\end{aligned} \tag{4.8}$$

From the equation above, we also denote $\overline{\boldsymbol{x}_{i,\boldsymbol{c}}} := [x^i_{1,\boldsymbol{c}},\ldots,x^i_{n,\boldsymbol{c}}]$ for the short notation. When two vectors c and c' are equivalent, i.e. they belong to the same interior of a face of Gröbner cone (Sturmfels & Thomas 1997), each sub-problem of evaluating the $\nu(S)$ has the same output vector $\boldsymbol{x}_{\boldsymbol{c}}(S)$. Hence, the final sum $\overline{\boldsymbol{x}_{i,\boldsymbol{c}}}$ is fixed in the chains of equalities (4.8) for each face of the Gröbner cone.

In the process of finding the Shapley value ϕ_i with the cost vector $\boldsymbol{c}$, we need to compute the value of vector $\overline{\boldsymbol{x}_{i,\boldsymbol{c}}}$. Afterwards, for any cost vector $\boldsymbol{c}'$ belonged to the same Gröbner cone of the cost function $\boldsymbol{c}$, we can reuse the value of $\overline{\boldsymbol{x}_{i,\boldsymbol{c}}}$ to compute the Shapley value of player i as $\phi_{i,\boldsymbol{c}'} = \boldsymbol{c}'^T\overline{\boldsymbol{x}_{i,\boldsymbol{c}'}} = \boldsymbol{c}'^T\overline{\boldsymbol{x}_{i,\boldsymbol{c}}}$.

□

Remark 4.4. For each equivalent class of cost function $\boldsymbol{c}$, we only need to evaluate the Shapley value $\phi_i^{\boldsymbol{c}}$ of player i once. In particular, the new cost vector $\boldsymbol{c}'$ can be checked if it stays in the same equivalent class with $\boldsymbol{c}$ by computing their Gröbner bases (Proposition 4.2). Afterwards, we can find the value of $\phi_i^{\boldsymbol{c}'}$ directly without recomputing the process.

Example 4.1. *We now show an example to illustrate the effective of theorem 4.3. The game can be expressed in the standard form as follows:*

$$r = 2, m = 4,\ \mathcal{A} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 5 & 10 & 25 \end{pmatrix},\ c = (1, 2, 1, 2),$$

and the resource matrix $\mathrm{B} = \begin{pmatrix} 7 & 9 & 11 & 13 & 15 \\ 36 & 55 & 71 & 48 & 92 \end{pmatrix}$ *of five players.*

The Gröbner basis of this problem is:

$$G_{\mathcal{A},\prec_{\boldsymbol{c}}} = \{(-5, 3, 4, -2), (-5, 6, 0, -1), (0, 3, -4, 1), (5, 0, -8, 3).\}$$

When we change the cost vector to $\boldsymbol{c}' := (1, 3, 2, 4)$. *We can inspect if the two cost vectors* $\boldsymbol{c}$ *and* $\boldsymbol{c}'$ *are equivalent by checking that the two Gröbner bases are the same* $G_{\mathcal{A},\prec_{\boldsymbol{c}}} = G_{\mathcal{A},<_{\boldsymbol{c}'}}$ *(Sturmfels & Thomas 1997, Proposition 3.2.).*

Using complete enumeration, we can find that $[x^i_{1,\boldsymbol{c}}, \ldots, x^i_{n,\boldsymbol{c}}] = [x^i_{1,\boldsymbol{c}'}, \ldots, x^i_{n,\boldsymbol{c}'}] = \overline{\boldsymbol{x}_{i,\boldsymbol{c}'}}$ *and is equal to:*

$$\overline{\boldsymbol{x}_{i,\boldsymbol{c}}} = \begin{pmatrix} 2.7500 & 3.8333 & 4.4167 & 8.9167 & 7.0833 \\ 0.8500 & 0.1000 & -0.1500 & 0.1000 & 0.1000 \\ 3.7333 & 5.0667 & 6.7333 & 4.0667 & 7.4000 \\ -0.3333 & 0 & 0 & -0.0833 & 0.4167 \end{pmatrix}$$

With the brute-force, the Shapley values of the corresponding games can be computed directly. For the cost vector $\boldsymbol{c}$, *the value* $\phi_{i,\boldsymbol{c}} = (7.52, 9.10, 10.85, 13.02, 15.52)$*; and with cost vector* $\boldsymbol{c}'$, *we have* $\phi_{i,\boldsymbol{c}'} = (11.43, 14.27, 17.43, 17.02, 23.85)$. *Afterwards, checking the equations of* $\phi_{i,\boldsymbol{c}} = \boldsymbol{c}^T\overline{\boldsymbol{x}_{i,\boldsymbol{c}}}$ *and* $\phi_{i,\boldsymbol{c}'} = \boldsymbol{c}'^T\overline{\boldsymbol{x}_{i,\boldsymbol{c}}}$ *confirms the result in theorem 4.3.*

4.4 Algebraic Gröbner Approach for the Shapley value of MIKG

This section demonstrates the way we incorporate the Algebraic Gröbner approach to calculating the Shapley value. This approach computes the Gröbner bases and test set only one time in advanced as a fixed cost and then employs the augmentation algorithm to solve $\nu(S)$ with different resource vectors $\boldsymbol{b}(S)$. We first describe the augmentation algorithm and then show how we can compute the Shapley value for the MIKG.

4.4.1 Augmentation algorithm for the multidimensional integer knapsack problem

The 4ti2 software is selected to compute the Gröbner basis and test set. Afterwards, augmentation algorithm to solve Integer Programming problems is the critical step of the AG approach. The algorithm is similar to the simplex method in the sense that the objective values improve through each iteration of the algorithms. Moreover, we can also apply the modified augmentation algorithm to find a feasible solution for the integer programming problem. The process of finding an optimal solution to $\mathrm{ILP}_{(\mathcal{A},c)}(\boldsymbol{b})$ via test sets (Loera et al. 2013) can be decomposed into the following steps:

1. Find an initial feasible solution $\boldsymbol{x}_0$ of $\mathrm{ILP}(\boldsymbol{b})$.
2. Decide whether $\boldsymbol{x}_0$ is optimal, and
3. If $\boldsymbol{x}_0$ is not optimal, find a better feasible solution.

In step 1, we can use the Gröbner basis to find out an initial feasible solution (for more detail, see A.3.3). This is a non-trivial problem in general, however, note that in the case of multidimensional integer knapsack games, the zero vector is one feasible solution. The next two steps will replace $\boldsymbol{x}_0$ by $\boldsymbol{x}_0 + \lambda \boldsymbol{t}$ where $(\lambda \in \mathbb{Z}, \boldsymbol{t} \in G_{\mathcal{A},\prec_c})$, and repeat until a better feasible solution of $f(\boldsymbol{x}) := \boldsymbol{c}^T\boldsymbol{x}$ is attained. If there is no such tuple $(\lambda, \boldsymbol{t})$, current point $\boldsymbol{x}_0$ is the optimal solution.

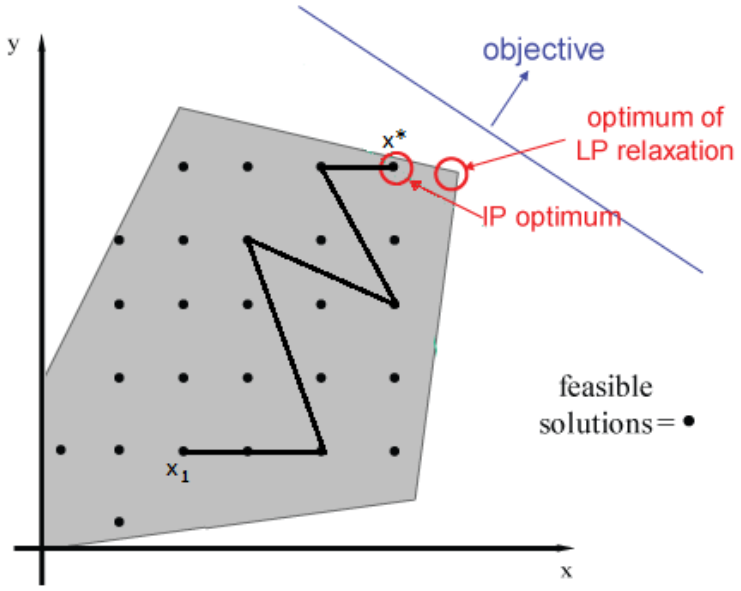

Figure 4.1: Augmentation algorithm for an example of Integer Program

To confirm the optimality condition for the current solution, we have to check all the vector $\boldsymbol{t}$ in the test set $G_{\mathcal{A},\prec_{\boldsymbol{c}}}$. Moreover, several test set vectors often can serve as a possible augmenting direction for a given non-optimal solution $\boldsymbol{x}_0$. Therefore, the question arises whether the augmenting vectors can be chosen in such a way that only a small number of augmentation steps are needed in order to reach an optimal solution.

Theorem 4.5. *Suppose that $\boldsymbol{x}_i$ is a current feasible solution at step i in the augmentation algorithm. If $\boldsymbol{x}_i$ is not an optimal solution, we can always improve the objective function by a vector in the Gröbner basis with step size one.*

In particular, if there is a constant λ such that $|\lambda| > 1$ and $f(\boldsymbol{x}_i + \lambda\boldsymbol{t}) > f(\boldsymbol{x}_i)$ then we have either the inequality $f(\boldsymbol{x}_i + \boldsymbol{t}) > f(\boldsymbol{x}_i)$ or the inequality $f(\boldsymbol{x}_i - \boldsymbol{t}) > f(\boldsymbol{x}_i)$.

Proof. We might assume that the the current solution $\boldsymbol{x}_i$ in the augmentation is not an optimal solution. Therefore there exists an vector $\boldsymbol{t}$ and $|\lambda| \geq 1$ such that vector $\boldsymbol{x}_i + \lambda\boldsymbol{t}$ is feasible and $f(\boldsymbol{x}_i + \lambda\boldsymbol{t}) > f(\boldsymbol{x}_i)$, or equivalently, $\boldsymbol{c}^T(\boldsymbol{x}_i + \lambda\boldsymbol{t}) > \boldsymbol{c}^T\boldsymbol{x}_i$. Because the MIKP model have the convex feasible space, either $\boldsymbol{x}_i + \boldsymbol{t}$ or $\boldsymbol{x}_i - \boldsymbol{t}$ is also satisfy all constraints in (4.5). Moreover, it follows from $\lambda\boldsymbol{c}^T\boldsymbol{t} > 0$, if $\lambda > 0$ then $\boldsymbol{c}^T(\boldsymbol{x}_i + \boldsymbol{t}) > \boldsymbol{c}^T\boldsymbol{x}_i$, otherwise if $\lambda < 0$, we take $\boldsymbol{t}' = -\boldsymbol{t}$ as the augmented vector. Hence, there must be an improving step size $\lambda = 1$. □

Remark 4.6. From the theorem 4.5, it is clear that the augmentation algorithm can search through all vector $\boldsymbol{t}$ in the Gröbner basis with a unit step size to find an improving solution in each iteration. In other words, the current solution x_k is optimal iff there exist no other improving direction when $\lambda = 1$.

This remark indicates that we theoretically can search over all the vector in the test set $G_{\mathcal{A},\prec_{\boldsymbol{c}}}$ with step size one for a better solution. However, for the current MIKP, there are a few better heuristic methods to select $\boldsymbol{t}$ and $\lambda_{\boldsymbol{t}}$ at the same time depending on the structure of the feasible area of $\mathrm{ILP}_{(\mathcal{A},\boldsymbol{c})}(\boldsymbol{b})$. Therefore, in the implementation of augmentation algorithm, there are two decisions of which values of λ and $\boldsymbol{t}$ to select for each iteration.

The first decision is how to choose a suitable vector $\boldsymbol{t} \in G_{\mathcal{A},\prec_{\boldsymbol{c}}}$, which we can utilise search algorithms such as Depth-first search (DFS) or Breath-first search (BFS). DFS and BFS are typical search algorithms used for graphs and trees. The decision to choose one over the other should be based on the type of problem instance that one is dealing with, in particular, it depends on the structure of the search tree and the number and location of optimal solutions. In tree searching, if our problem is to search the solution that is more likely to close to the tree root, we would prefer BFS, and if the target node is close to a leaf, we would prefer DFS. In our case, after making the experimental

comparison between the two methods, using the DFS could generate a large number of small steps to finally arrive at the optimal solution. Hence, we decide to choose the BFS, in which check the candidates from all the possible vectors in the Gröbner basis to find the best solution in the algorithm 3.

The second decision is how we can search the integer value of λ to maximise $f(\boldsymbol{x}_0+\lambda\boldsymbol{t}) := \boldsymbol{c}^T(\boldsymbol{x}_0 + \lambda\boldsymbol{t})$ such that the point $\boldsymbol{x}_0 + \lambda\boldsymbol{t}$ is feasible. We notice that since $\mathcal{A}x_0 = \boldsymbol{b}$ and $\mathcal{A}\boldsymbol{t} = 0$, the next point satisfies $\mathcal{A}(x_0 + \lambda\boldsymbol{t}) = \boldsymbol{b}$, for $\forall\lambda \in \mathbb{Z}$ and $\boldsymbol{t} \in G_{\mathcal{A},\prec_{\boldsymbol{c}}}$. Therefore, from the problem (4.7), the feasibility condition of $\boldsymbol{x}_0 + \lambda\boldsymbol{t}$ means all of its coordinators must not be negative.

For this step, we need to solve a constrained optimisation problem with only one integer variable $\lambda \in \mathbb{Z}$. Let $\boldsymbol{t} = \{t^i\}_{i=1}^n$ and $\boldsymbol{x}_0 = \{\boldsymbol{x}_0^i\}_{i=1}^n$ be the two fixed vectors in $G_{\mathcal{A},\prec_{\boldsymbol{c}}}$ and $\mathbb{Z}^n$. The process will compute $\lambda_{\boldsymbol{t}} = \underset{\lambda\in\mathbb{Z}}{\operatorname{argmax}} f(\boldsymbol{x}_0 + \lambda\boldsymbol{t}) = \underset{\lambda\in\mathbb{Z}}{\operatorname{argmax}} \{\boldsymbol{c}^T\boldsymbol{x}_0 + \lambda\boldsymbol{c}^T\boldsymbol{t}\}$ such that $\boldsymbol{x}_0^i + \lambda.t^i \geq 0$ for $\forall i = \{1, \ldots, n\}$. With the particular coordinator values of vector $\boldsymbol{t}$ and $\boldsymbol{x}_0$, we can design a simple heuristic method to find the step size $\lambda_{\boldsymbol{t}} = \underset{\lambda\in\mathbb{Z}}{\operatorname{argmax}} \lambda\boldsymbol{c}^T\boldsymbol{t}$.

We now show our modified augmentation algorithm in the detail below:

Algorithm 3 : Augmentation Algorithm with the rhs vector $\boldsymbol{b}$

Input: - $\mathcal{A}, \boldsymbol{b}, \boldsymbol{c}$ where $\mathcal{A}$ is the matrix, $\boldsymbol{b}$ is rhs, and vector $\boldsymbol{c}$ give a comparison order,
- A test set $G_{\mathcal{A},\prec_{\boldsymbol{c}}}$ of the ILP(4.7) (using 4ti2 software),
- A feasible solution $\boldsymbol{x}_0 = \boldsymbol{0}$ to ILP($\boldsymbol{b}$).
Output: an optimum $\boldsymbol{x}^*$ of ILP($\boldsymbol{b}$)
Initialize: $f_s = 0$, $opt = 0$ # the optimality variable
while $opt = 0$ **do**
  for $\forall\boldsymbol{t} \in G_{\mathcal{A},\prec_{\boldsymbol{c}}}$ **do**
    Find $\lambda_{\boldsymbol{t}} = \underset{\lambda\in\mathbb{Z}}{\operatorname{argmax}} f(\boldsymbol{x}_0 + \lambda\boldsymbol{t})$ such that $\boldsymbol{x}_0 + \lambda\boldsymbol{t}$ feasible.
    Set $f_{\boldsymbol{t}} = f(\boldsymbol{x}_0 + \boldsymbol{t}\lambda_{\boldsymbol{t}})$
  end for
  # Find the next $(\boldsymbol{t}, \lambda)$ by the BFS method
  Compute $(\boldsymbol{t}_e, \lambda_e) = \underset{\boldsymbol{t}\in G_{\mathcal{A},\prec_{\boldsymbol{c}}}}{\operatorname{argmax}} f_{\boldsymbol{t}}$; and set $\boldsymbol{x}_0 = \boldsymbol{x}_0 + \boldsymbol{t}_e\lambda_e$, $f_e = f(\boldsymbol{x}_0)$
  # Checking the optimality
  if $f_e = f_s$ **then**
    $opt = 1$,
  else
    $f_s = f_e$,
  end if
end while
return $\boldsymbol{x}^* = \boldsymbol{x}_0$

To combine the previous results and the computation of the Shapley value, we have the following theorem.

Theorem 4.7. *Suppose that the value $\nu(S)$ of problem (4.7) is bounded for $\forall S \subseteq N$. For solving the Integer Program of the value function of coalition S, let us assume that the coefficients of $(\mathcal{A}, \boldsymbol{b}(S), \boldsymbol{c})$ are all integers, and the Gröbner basis $G_{\mathcal{A},\prec_c}$ is computed in advance. Then the number of steps in the augmentation algorithm for solving the ILP problem (4.7) is bounded by $\nu(S)$.*

Proof. With the augmentation algorithm, it continues the iteration when each step will improve the outcome at least one. This process is done by searching through all the vectors in the Gröbner basis for the improvement. Therefore, maximum number of steps is $\nu(S)$ for the augmentation algorithm. □

Therefore, given the input data as the parameter tuples $\{\mathcal{A}, \mathrm{B}, c\}$ and the Gröbner basis $G_{\mathcal{A},\prec_c}$, from theorem 4.7 and $\nu(S) \leq \nu(N),\ \forall S \subset N$, we can imply that the value of $\nu(S)$ can be computed in polynomial time. A similar complexity result to the Gröbner basis method is shown in (Onn 2010, Theorem 3.12) about the complexity of the Graver basis method when solving the integer programming problems.

4.4.2 Framework for finding the exact Shapley value of MIKG

The Shapley value is based on the notion that the allocation for a player should be proportional to that player's contribution to the game, i.e., how much value that player creates. Therefore, we have to find all the coalition values $\nu(S), \forall S \subseteq N$ for the exact Shapley value. If we can compute $\nu(S)$ quicker for each S, the total computational time will decrease significantly.

In the Algebraic Gröbner approach, the Gröbner test set $G_{\mathcal{A},\prec_c}$ is computed in advanced with only two parameters of matrix $\mathcal{A}$ and cost vector $\boldsymbol{c}$. Afterwards, the Augmentation Algorithm 3 solves the $\mathrm{ILP}_{(\mathcal{A},\boldsymbol{c})}(\boldsymbol{b})$ problem (4.7) for each coalition S, this can be done substantially faster compared to other current methods. Since we only need to change the rhs $\boldsymbol{b}$ of the model, the same Gröbner test set is applied to find all coalition values.

We now show the AG approach to compute the Shapley value in Algorithm 4.

In general, the Algorithm 4 is able to compute the real Shapley value for the MIKG game with a small number of players (i.e. less than 20, seen the Table 4.2). This process requires the evaluation of 2^n values of the coalition, each of which is an integer optimisation problem. However, when the number of players increases we apply the sampling-based technique to approximate the Shapley value with a limited number of samples.

Algorithm 4 : Shapley value - Algebraic Gröbner Exact Algorithm

Input: Problem instance $(\mathcal{A}, \mathrm{B}, \boldsymbol{c})$ with the size vector (r, q, n).
Output: Exact Shapley value for each player i.
Initialize: Find the Gröbner test set $G_{\mathcal{A},\prec_{\boldsymbol{c}}}$ of matrix $\mathcal{A}$ and the term order $\prec_{\boldsymbol{c}}$.
Build the Integer Program model with the parameters $(\mathcal{A}, \boldsymbol{c}, \boldsymbol{b})$ in problem (4.7).
for all coalition $S \subseteq N$ **do**
  Solve the $\mathrm{ILP}_{(\mathcal{A},\boldsymbol{c})}(\boldsymbol{b}(S))$ by Augmentation Algorithm 3 to find $\nu(S)$.
end for
for i =1:n **do**
  Calculate the Shapley value $\boldsymbol{\phi}_i$ by equation (3.3)
end for

4.4.3 Algebraic Gröbner Sampling method for approximating the Shapley value of large MIKG

The sampling-based technique is applied and shown the effect when we approximate the Shapley value for large-scale MIKG. In these games, instead of solving all 2^n integer programs, we utilise a randomised method based on simple random sampling technique to restrict the sampling budget. Hence, we only solve the ILP problems of a limited number of samples to find the coalition values. The number of samples taken to approximate the Shapley value depends on the level of exactness in required time.

By simple random sampling (SRS) technique, we sample a random permutation π of all players N and find the coalition S of the players arriving before i in this permutation. The marginal contribution of the player i into coalition S is $\nu(S \cup \{i\}) - \nu(S)$. From the equations (3.3), the average value of all these marginal contributions will converge to the Shapley value of player i. We describe the combination of SRS approximation and the algebraic Gröbner approach in the following algorithm:

The algebraic Gröbner approach required us to find the Gröbner bases for our problems in the beginning as fixed cost. However, the augmentation algorithm step is usually faster than the typical ILP solver. Therefore, if we have to solve the Integer Programming model many times for a specific problem model with changed rhs, this process with smaller variable costs brings down the total calculation time in the long run.

4.5 Computational Experiments

The algorithms have been implemented in Matlab and run on an Intel-core i5 PC with 2.6 GHz CPU and 4GB RAM. In the experiments, we utilise the program 4ti2 to compute the Gröbner bases. After the Gröbner basis computational times for some test instances are shown, we check our AG algorithm into those instances. We test the algorithm 4 with

Algorithm 5 : Shapley value - Algebraic Gröbner Sampling Algorithm

Input: - Problem instance $(\mathcal{A}, \mathrm{B}, \boldsymbol{c})$ with the size vector (r, q, n),
- A sampling budget κ for each player.
Output: Approximated Shapley value $\boldsymbol{\phi}_i$ for each player i.
Initialize: - Let the grand coalition be $N = (1, \ldots, n)$, and sample count $j = 0$;
- The marginal contribution sum $\varrho(i) = 0$ for each $i \in \{1, \ldots, n\}$;
- Calculate Gröbner test set $G_{\mathcal{A}, \prec_{\boldsymbol{c}}}$ of matrix $\mathcal{A}$ and the term order $\prec_{\boldsymbol{c}}$;
- Build the Integer Program model with the parameters $(\mathcal{A}, \boldsymbol{b}, \boldsymbol{c})$ in problem (4.7).
while $j < \kappa$ **do**
  Randomly select a permutation π of N;
  for $i = 1 : n$ **do**
    Find the coalition $S := Pre^i(\pi)$ which arrived before i in π.
    Construct two resource vectors $\boldsymbol{b}(S)$ and $\boldsymbol{b}(S \cup \{i\})$
    Compute $\nu(S)$ and $\nu(S \cup \{i\})$ by apply the Augmentation Algorithm 3.
    $\varrho(i) = \varrho(i) + \{\nu(S \cup \{i\}) - \nu(S)\})$;
  end for
  j= j +1;
end while
for $i = 1 : n$ **do**
  Approximate the Shapley value $\boldsymbol{\phi}_i = \varrho(i)/\kappa$ from the formula (3.3)
end for

the small-scale instance where the number of players less than 20, and the algorithm 5 otherwise.

First, we show the computational time and the numbers of vectors in the Gröbner bases for some ILP test instances in Table 4.1. The data in the test problems has been randomly generated from uniform distribution for the integral matrices W of various sizes (ranging from 4×6 to 11×15) and the matrices' entries of magnitude from 0 to k, where $k = 100, 20, 10, 1$. The linear cost function has the coefficient $c_{ij} \in [0, 50]$, and the entries of integer matrix B are in the range from 0 to 200. For example, the test instance mat$r \times q - \mathrm{e}\{k\}$ shows the calculation time of the Gröbner basis for a knapsack problem with q objects in dimension r, such that the entries in the underlying matrix are generated from the set $\{0, 1, \ldots, k\}$. The column N_{GB} shows the number of vectors in the Gröbner basis and the column T_{GB} presents the computational time of these bases in seconds. The Table 4.1 shows that the growth of the computational time and the size of the Gröbner bases increase quickly for larger instances using the 4ti2 software.

The table indicates Gröbner basis computational time is highly sensitive to the number of variables (# columns and # rows of matrix $\mathcal{A}$) and the magnitude of the integral entries. In particular, the Gröbner bases can be calculated under 5 seconds for the size of matrix W less than 11×13. When the problem size gets larger, the total time of solving an ILP problem is the sum of the fixed cost of computing the Gröbner bases and the variable cost of implementing the augmentation algorithm. Therefore, if we solve

Table 4.1: The computational times and sizes of Gröbner bases for different MIK instances

$TestCases$	N_{GB}	T_{GB}	$TestCases$	N_{GB}	T_{GB}
mat4 $\times$ 6 $-$ e100	80	0	mat5 $\times$ 7 $-$ e100	84	0
mat4 $\times$ 6 $-$ e20	65	0	mat5 $\times$ 7 $-$ e20	80	0
mat4 $\times$ 6 $-$ e10	55	0	mat5 $\times$ 7 $-$ e10	53	0
mat4 $\times$ 6 $-$ e1	6	0	mat5 $\times$ 7 $-$ e1	8	0
mat4 $\times$ 8 $-$ e100	238	0.01	mat5 $\times$ 9 $-$ e100	422	0
mat4 $\times$ 8 $-$ e20	175	0	mat5 $\times$ 9 $-$ e20	357	0.01
mat4 $\times$ 8 $-$ e10	127	0	mat5 $\times$ 9 $-$ e10	259	0
mat4 $\times$ 8 $-$ e1	8	0	mat5 $\times$ 9 $-$ e1	12	0
mat6 $\times$ 8 $-$ e100	210	0.01	mat7 $\times$ 9 $-$ e100	319	0.01
mat6 $\times$ 8 $-$ e20	126	0	mat7 $\times$ 9 $-$ e20	246	0.01
mat6 $\times$ 8 $-$ e10	68	0	mat7 $\times$ 9 $-$ e10	155	0
mat6 $\times$ 8 $-$ e1	10	0	mat7 $\times$ 9 $-$ e1	9	0
mat6 $\times$ 10 $-$ e100	216	0.01	mat7 $\times$ 11 $-$ e100	440	0
mat6 $\times$ 10 $-$ e20	47	0.01	mat7 $\times$ 11 $-$ e20	362	0.01
mat6 $\times$ 10 $-$ e10	37	0	mat7 $\times$ 11 $-$ e10	317	0
mat6 $\times$ 10 $-$ e1	10	0	mat7 $\times$ 11 $-$ e1	13	0
mat8 $\times$ 10 $-$ e100	1840	0.19	mat9 $\times$ 11 $-$ e100	7219	2.38
mat8 $\times$ 10 $-$ e20	1405	0.14	mat9 $\times$ 11 $-$ e20	5085	1.36
mat8 $\times$ 10 $-$ e10	755	0.03	mat9 $\times$ 11 $-$ e10	3816	0.92
mat8 $\times$ 10 $-$ e1	16	0	mat9 $\times$ 11 $-$ e1	37	0
mat8 $\times$ 12 $-$ e100	3883	0.73	mat9 $\times$ 13 $-$ e100	3463	0.47
mat8 $\times$ 12 $-$ e20	2387	0.34	mat9 $\times$ 13 $-$ e20	2751	0.38
mat8 $\times$ 12 $-$ e10	1212	0.11	mat9 $\times$ 13 $-$ e10	1593	0.12
mat8 $\times$ 12 $-$ e1	15	0	mat9 $\times$ 13 $-$ e1	35	0
mat10 $\times$ 12 $-$ e100	5030	1.26	mat11 $\times$ 13 $-$ e100	*	*
mat10 $\times$ 12 $-$ e20	3893	0.86	mat11 $\times$ 13 $-$ e20	252951	6988.7
mat10 $\times$ 12 $-$ e10	2252	0.27	mat11 $\times$ 13 $-$ e10	179443	3537.4
mat10 $\times$ 12 $-$ e1	47	0	mat11 $\times$ 13 $-$ e1	156	0
mat10 $\times$ 14 $-$ e100	*	*	mat11 $\times$ 15 $-$ e100	*	*
mat10 $\times$ 14 $-$ e20	124648	1342.5	mat11 $\times$ 15 $-$ e20	*	*
mat10 $\times$ 14 $-$ e10	73820	453.8	mat11 $\times$ 15 $-$ e10	221660	3699
mat10 $\times$ 14 $-$ e1	107	0.01	mat11 $\times$ 15 $-$ e1	330	0.02

only the ILP once or a few times, it may seem not worth the effort to invest in the computation of Gröbner bases. However, in the Shapley value formula, we need to solve ILP for a large number of times with the fixed underlying structure except the varied rhs. We also notice that these parameters are also fairly depended on the magnitude of matrice entries. Hence, for the problem with smaller entry values, it will be easier to apply our approach.

In next part, we apply our proposed method to compute the exact Shapley values for small-scale problem instance when the number of players is less than 20. For larger cases with $n = 20, 40, 60, 80, 100$, the algebraic Gröbner algorithm 5 is compared with the integer programming solver CPLEX 12.6 where the performance measure is the computational time.

4.5.1 Illustrative example

The next example shows us how to utilise the 4ti2 package, which is an open source package to calculate the Gröbner bases, to help solve a simple integer program.

Example 4.2. ***(Agebraic Gröbner method for Linear Integer Problem)***

Consider a multidimensional knapsack problem of the following form:

$$\max\ \boldsymbol{p}^T\boldsymbol{z} \quad s.t.\ \ \mathrm{W}\boldsymbol{z} \leq \boldsymbol{b}, \quad \boldsymbol{z} \in \mathbb{Z}_+^8,$$

where $\boldsymbol{b} = [433, 432, 344, 619, 442, 508]^T$, $\boldsymbol{p} = [3, 9, 31, 28, 21, 8, 10, 7]^T$, *and the weighted matrix with eight-item types and six dimensions (constraints)*

$$\mathrm{W} = \begin{pmatrix} 82 & -28 & 96 & 80 & 68 & 71 & 70 & 77 \\ 91 & 55 & 49 & 96 & 76 & -24 & -32 & 80 \\ 13 & 96 & 81 & 66 & 75 & 28 & 96 & 19 \\ 92 & 97 & 15 & -4 & 40 & 5 & -16 & 49 \\ 64 & -16 & 43 & 85 & 66 & 10 & 44 & 45 \\ 10 & 98 & 92 & 94 & -18 & 83 & 39 & 65 \end{pmatrix}.$$

First, we transform the problem into the standard form with new slack variables $\boldsymbol{y} \in \mathbb{Z}_+^6$ *and the new coefficients of* $\mathcal{A} = [\mathrm{W}|I_6]$ *and* $\boldsymbol{c} = [\boldsymbol{p}|0_6]$. *With new variable* $\boldsymbol{z} = [\boldsymbol{x}, \boldsymbol{y}]$, *we have the problem structure as:*

$$\max\ \boldsymbol{c}^T\boldsymbol{x} \quad s.t.\ \ \mathcal{A}\boldsymbol{x} = \boldsymbol{b}, \quad \boldsymbol{x} \in \mathbb{Z}_+^{14}$$

Using 4ti2 *software, the computational time of this Gröbner basis is less than* 0.03 *seconds and it contains* 1057 *vectors of dimension* 14.

$$G_{\mathcal{A},\prec_c} = \left\{ \begin{array}{cccccccccccccc} -5 & 0 & 2 & -2 & 0 & 0 & 0 & 1 & 301 & 469 & 16 & 373 & 359 & -11 \\ -4 & 0 & 1 & -1 & 0 & 0 & 0 & 1 & 235 & 331 & 18 & 300 & 253 & -23 \\ \vdots & & & & & & & & & & & & & \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 16 & -47 & 15 & 19 & -42 & -2 \\ \vdots & & & & & & & & & & & & & \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 80 & 96 & 66 & -4 & 85 & 94 \\ \vdots & & & & & & & & & & & & & \\ 0 & 0 & 2 & -2 & 0 & -1 & 0 & 0 & 39 & 70 & -2 & -33 & 94 & 87 \\ \vdots & & & & & & & & & & & & & \\ 5 & 0 & 1 & -1 & -1 & 0 & 0 & 0 & -358 & -332 & -5 & -439 & -212 & -66 \\ 6 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -424 & -470 & -3 & -512 & -318 & -78 \end{array} \right\},$$

The straightforward initial feasible solution is

$$\boldsymbol{x}_0 = [0,0,0,0,0,0,0,0,433,432,344,619,442,508].$$

Afterwards, we move along the vector

$$\boldsymbol{t}_1 = [0,0,0,-1,0,0,0,0,80,96,66,-4,85,94]$$

with the step length $\lambda_1 = -4$ *to get the next feasible solution :*

$$\boldsymbol{x}_1 = [0,0,0,4,0,0,0,0,113,48,80,635,102,132].$$

Repeating this process we have:

$$\boldsymbol{t}_2 = [0,0,-1,1,0,0,0,0,16,-47,15,19,-42,-2] \text{ and } \lambda_2 = -4.$$

$$\boldsymbol{x}_2 = [0,0,4,0,0,0,0,0,49,236,20,559,270,140],$$

$$\boldsymbol{t}_3 = [0,0,2,-2,0,-1,0,0,39,70,-2,-33,94,87] \text{ and } \lambda_3 = -1.$$

Finally

$$\boldsymbol{x}^* = [0,0,2,2,0,1,0,0,10,166,22,592,176,53].$$

The final solution of the optimisation problem is $\boldsymbol{x}^* = [0,0,2,2,0,1,0,0]$ *with the optimal value is* 126. *Using the algebraic Gröbner method, this ILP problem is solved in 0.019 seconds by the augmentation algorithm, however, CPLEX need* 0.1 *seconds. In the next section, we demonstrate the way to apply this algebraic approach to reducing the total computation time of the Shapley value.*

For the game version of this knapsack problem with ten players, the MIKG is constructed by its resource vector matrix B. *The resource vectors of these players, which entries are randomly generated in the range* $[0, 300]$*, are as follows:*

$$\mathrm{B} = [\boldsymbol{b}_1, \ldots . \boldsymbol{b}_{10}] = \begin{pmatrix} 198 & 57 & 27 & 65 & 124 & 82 & 24 & 208 & 223 & 218 \\ 275 & 218 & 147 & 53 & 14 & 196 & 288 & 269 & 47 & 41 \\ 190 & 236 & 279 & 22 & 63 & 287 & 162 & 102 & 209 & 94 \\ 110 & 292 & 237 & 268 & 219 & 131 & 252 & 19 & 253 & 126 \\ 166 & 256 & 146 & 192 & 195 & 21 & 51 & 260 & 218 & 264 \\ 59 & 163 & 137 & 43 & 144 & 17 & 78 & 87 & 108 & 46 \end{pmatrix}.$$

The Shapley value is $(65.34, 48.12, 40.10, 17.60, 34.79, 43.53, 32.41, 57.76, 64.59, 46.76)$ *and the total payoff of the grand coalition is* 451*. The time of calculating exact algorithm using the CPLEX is 104.27(s), but using the Gröbner method, it took only 16.39(s). This small example show the capability of the algebraic Gröbner approach in solving Integer Program problems and computing the Shapley value of MIKG.*

In our MIKG model, the algebraic Gröbner approach has the advantage over other methods to solve ILP sensitivity analysis when changing the rhs of the ILP formulation. Since the Gröbner bases is calculated at the beginning with a fixed cost, the augmentation algorithm can solve $c(S)$ with less computational time compared to other solvers such as IBM CPLEX and MATLAB intlinprog. In the next part, we present the computational experiments with some test instances of MIKG.

4.5.2 Exact Shapley value calculation for MIKG

Computations of the Shapley value is known as $\#P$-hard problems for many cooperative games. However, in the cases with the number of players $n <= 20$, we are able to find the exact Shapley value if there is enough time to calculate all the 2^n characteristic values. Table 4.2 show this in detail where the problem sizes (r, q) range from 4×6 to 10×12 and the numbers of players range from 5 to 17. It also shows the time to compute the Gröbner bases in $T_{Groebner}$ column and the computational time of augmentation step in T_{Augmt} column. The total computational times of Algebraic Gröbner algorithm 4 in column T_{AG} and the time of CPLEX in column T_{CPLEX}.

At the beginning, the algebraic approach required us to find the Gröbner basis for the matrix in standard form $\mathcal{A} = [\mathrm{W}|\mathrm{I}_r]$ as fixed cost. However, the step of solving the integer programs with the augmentation algorithm is quicker than the typical ILP solver. As we need to solve the Integer Programming model many times for a specific problem instance with changed rhs, the algebraic process with low variable costs will

$S_{Problem}$	$T_{Groebner}$	$N_{Players}$	T_{Augmt}	T_{AG}	T_{CLPEX}
$mat4 \times 6 - e100$	0	5	0.06	0.06	0.29
		8	0.22	0.22	3.31
		11	8.07	8.07	62.7
		14	20.57	20.57	259.08
		17	185.17	185.17	1276.62
$mat4 \times 8 - e10$	0	5	0.05	0.06	1.35
		8	0.26	0.26	11.35
		11	2.47	2.47	23.26
		14	22.01	22.01	227.33
		17	193.14	193.14	1541.29
$mat6 \times 8 - e100$	0	5	0.06	0.06	0.42
		8	0.26	0.27	4.29
		11	13.19	13.20	192.75
		14	23.04	23.05	422.17
		17	202.40	202.41	2413.80
$mat6 \times 10 - e10$	0.01	5	0.09	0.1	0.43
		8	0.21	0.21	2.27
		11	1.95	1.95	17.76
		14	18.67	18.67	201.48
		17	172.05	172.05	1440.58
$mat8 \times 10 - e100$	0.19	5	0.26	0.45	0.51
		8	1.19	1.38	4.59
		11	15.13	15.32	154.47
		14	63.85	64.04	462.63
		17	533.26	533.45	2356.87
$mat8 \times 12 - e10$	0.05	5	0.46	0.51	0.88
		8	0.80	0.85	4.59
		11	7.09	7.14	37.30
		14	54.63	54.68	253.64
		17	482.25	482.30	2003.28
$mat10 \times 12 - e100$	1.26	5	0.38	1.64	0.63
		8	2.53	3.79	5.77
		11	21.07	22.33	237.75
		14	182.94	184.2	636.16
		17	847.89	849.15	4721.02

Table 4.2: Comparison of the computational time to find the exact Shapley value

bring down the total calculation time in the long run. Therefore, for most of the tested instances in Table 4.2, the computational times of AG method is significantly better than the CPLEX solver. The AG method is competitive to CPLEX, only when the size of the matrix $\mathcal{A}$ is not too large, since it might take a large amount of time to calculate the Gröbner basis.

4.5.3 Approximating the Shapley value for large scale MIKG

When the number of players is getting larger, the algebraic Gröbner sampling algorithm 5 will be implemented to compared with CPLEX solvers. We fix the sampling budget (the number of samples is 100 for each Shapley value of each player) for sampling. We can apply the AG method to find the approximation Shapley value and compare with the exact Shapley value to see the potential errors. For some problem instances $(\mathcal{A}, \mathrm{B}, \boldsymbol{c})$, we consider the case when the number of players ranges from 20 to 100.

From the numerical result, we found that the algorithm was competitive to CPLEX, often faster than five times for some problem instances. The Table 4.3 shows the computational times of the AG methods compared to CPLEX for solving Integer Programs. Since both solvers are combined with the sampling technique to approximate the Shapley value, we can observe the advantages of the algebraic Gröbner method. The computational time of Gröbner basis will increase quickly when the sizes or entries of the matrix $\mathcal{A}$ grow (Table 4.1), however, we only need to compute it once in advanced. Afterwards, the Tables 4.2 and 4.3 indicate that the augmentation algorithm will solve Integer Programs quicker than CPLEX for most cases.

In the next part, the errors of our sampling techniques are investigated. Considering the cases when there are only two types of players, we can evaluate the exact Shapley value for this type of MIKGs with large number of player (for example, $n = 100$). Afterwards, we apply the algorithm 5 to compare with the exact algorithm to see the errors (RMSE and MAPE). The *mean absolute percentage error* (MAPE) is a measure between the approximate and the exact Shapley values, and which is calculated as: $\mathrm{MAPE}(\boldsymbol{\phi}, \widehat{\boldsymbol{\phi}}) = 100.\dfrac{1}{n}\displaystyle\sum_{i=1}^{n} \left|\dfrac{\widehat{\phi}_i - \phi_i}{\phi_i}\right|$, and the *root mean squared error* (RMSE) is a quadratic scoring rule which measures the average magnitude of the error. The RMSE is calculated as: $\mathrm{RMSE}(\boldsymbol{\phi}, \widehat{\boldsymbol{\phi}}) = \sqrt{\dfrac{1}{n}\displaystyle\sum_{i=1}^{n}(\widehat{\phi}_i - \phi_i)^2}$, where $\boldsymbol{\phi}$ is the real Shapley value and $\widehat{\boldsymbol{\phi}}$ is the approximate Shapley value. We want to combine both sampling errors MAPE and RMSE because the MAPE depends on the proportional value of the output, and the RMSE relies on the total vector value $\nu(N)$.

The Figures 4.2 and 4.3 show the relationship between the timing budget (in seconds) and the MAPE sampling errors. We also see that the two methods of using the Algebraic Gröbner approach and the CPLEX solver generate different errors with respect to the time.

From the graph for most of our games with 100 players, we see that if we set the time limit is less than one hour, the sampling errors are quite small (i.e. the RMSE less than

$S_{Problem}$	$T_{Groebner}$	$N_{Players}$	T_{Augmt}	T_{AG}	T_{CLPEX}
$mat5 \times 7 - e100$	0	20	1.15	1.15	37.16
		40	1.58	1.58	86.34
		60	2.47	2.47	162.77
		80	3.16	3.16	198.08
		100	4.66	4.66	211.39
$mat5 \times 9 - e100$	0	20	0.54	0.54	7.52
		40	1.23	1.23	14.40
		60	1.84	1.84	27.21
		80	2.75	2.75	48.42
		100	3.78	3.78	67.58
$mat7 \times 9 - e20$	0.01	20	2.53	2.54	36.55
		40	4.39	4.40	64.14
		60	7.23	7.24	98.82
		80	9.72	9.73	140.93
		100	13.01	13.02	209.10
$mat7 \times 11 - e20$	0.01	20	2.97	2.98	40.08
		40	6.19	6.20	62.16
		60	9.79	9.79	98.06
		80	13.35	13.35	127.54
		100	16.77	16.77	161.93
$mat9 \times 11 - e10$	0.92	20	4.23	5.15	7.42
		40	10.99	11.91	13.98
		60	17.15	18.07	18.75
		80	22.86	23.78	35.51
		100	28.35	29.27	73.28
$mat9 \times 13 - e10$	0.09	20	19.84	19.93	50.57
		40	31.33	31.42	105.32
		60	50.14	50.23	164.09
		80	65.20	65.29	185.46
		100	86.15	86.24	288.82
$mat11 \times 13 - e1$	0.02	20	4.32	4.34	59.08
		40	6.19	6.21	64.81
		60	7.72	7.74	90.97
		80	10.47	10.49	100.77
		100	13.55	13.57	134.95
$mat11 \times 15 - e1$	0	20	6.72	6.72	17.41
		40	9.67	9.67	32.58
		60	14.95	14.95	43.99
		80	19.68	19.68	57.76
		100	26.90	26.90	71.92

Table 4.3: Time comparison of algebraic Gröbner method and CPLEX solver for approximation method to calculate the Shapley value with 100 samples for each player

2 for $\nu(N) = 2671$ and the MAPE is less than 1%). Under the same time limit, the AG approach can solve the Shapley value with significantly smaller errors compared to the

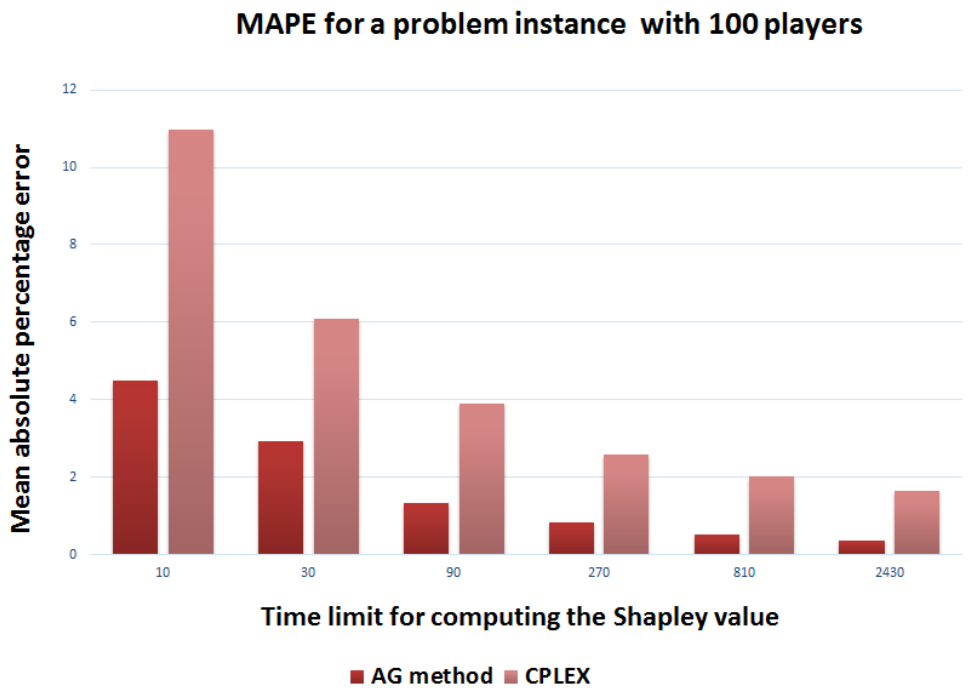

Figure 4.2: MAPE comparison of AG method and CPLEX on the test instance with 100 players

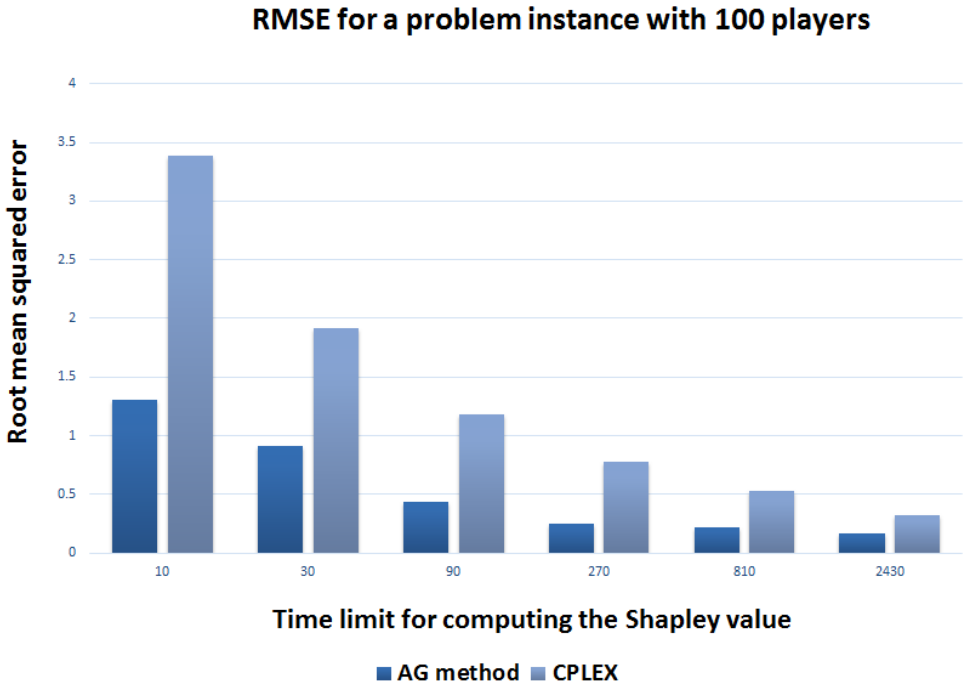

Figure 4.3: RMSE comparison of AG method and CPLEX on the test instance with 100 players

CPLEX solver for the tested instances.

4.6 Conclusion

We have introduced the Multidimensional Integer Knapsack Games and its applications. The Shapley value is proposed to allocate the total value of the collaboration for the grand coalition of players. We proposed an algebraic Gröbner approach to finding the Shapley values of the MIKGs. For the games with a large number of players ($n \geq 20$), the Monte Carlo sampling method is used to combine with the algebraic Gröbner approach. The sampling technique reduces the number of coalition values that we have to compute without scarifying too much exactness in the approximation of the Shapley value. We also summary some results of the experiments and show the effects of our algorithm in different problem instances.

Chapter 5

Discussions and Conclusions

In this thesis, the aim was to integrate approaches from operational research, computer science, and cooperative game theory to allocate the payoff (cost/yield) to a coalition of players in several operational research games. Our methods are not only applicable to a small group of players with similar contributions, but it is relevant for large cooperative groups of diverse participants. The thesis constructs some novel classes of operational research games and shows their practical applications, where the underlying characteristic functions of these cooperative games are described by mathematical programming problems. This algorithmic perspective of cooperative game theory is promising to play a greater role in many industries as various game theoretic concepts are utilised in the major collaborative networks.

5.1 Summary of Research Contributions

Main contributions of the thesis range from several new classes of cooperative games and theoretical properties of their solution concepts to the implementation of novel algorithms and the analyses of these payoff allocations to many practical applications.

In the thesis, we have introduced two classes of cooperative games including the Generalised Minimum Spanning Tree Game and the Multidimensional Integer Knapsack Game. The former class of cooperative games is the generalised version of the Minimum-cost Spanning Tree Game for network with multi-cluster structure (Section 2.2). The least-core of GMSTG is shown to have several interesting properties similar to those of the original MSTG core. The latter MIKG class assembles some operational research

games with the characteristic function having integer programming structure and super-additive property (Section 4.3). We also compared the MIKG model with other cooperative games in the literature, such as the Coalitional Resource Games and the Knapsack Budget Game.

In cooperative game theory, the core and least core are the basic solution concepts due to their properties of stability. In general, the players sharing the payoff allocation by the core/least core do not have the incentive to deviate from the collaboration. Although the core might be empty for some operational research games, e.g. the case of GMSTG in Example 2.1, the least core always exists. To find a payoff allocation in the least core, we employed the constraint generation approach for the GMSTG (Section 2.3.2). This method has an advantage of computing a least core allocation effectively without knowing all the coalition values. Several theoretical results and illustrative examples are given to show the relationship of the cores of the GMSTG and other related games (Section 2.2.3).

The Shapley value is another popular solution concept related to the fairness property in cooperative game theory. In this thesis, we applied the Shapley value for the linear production games and the super-additive integer optimisation games (in particular, the MIKG). Finding the exact Shapley values for these operational research games are quite challenging when the games have large numbers of players. In these situations, we decided to approximate the Shapley values by combining the sampling approach in Monte Carlo simulation and some sensitivity analysis techniques of mathematical programming. In the case of LPG, the linear programming sensitivity analysis helps computing the marginal contribution of a player to a coalition without starting from scratch each time (Section 3.3). In the case of MIKG, we employed the algebraic Gröbner method to generate a test set for the underlying Integer Program, hence, reduced the computational time of the coalition valuations (Section 4.4). For both types of games, the computational results showed the effectiveness of these methods for several test instances where there are up to 100 players. In the literature of cooperative games, although many optimisation techniques have been employed to find payoff allocations, these approaches are the first to apply LPSA and Gröbener bases for computing the Shapley value of operational research games with many players.

5.2 Limitations of the Research Results

In this part, we will mention some limitations of our research. First, there are some other types of operational research games which we do not work on such as the inventory games and sequencing games, which are motivated by the related operational research problems.

Another type of operational research games, which we do not mention in this work, is the non-linear integer optimisation game. In the game, a coalition value is the outcome of the corresponding nonlinear convex optimisation problems. The non-linearity character could appear in either the constraints or objective function of the coalition formulation. Since solving the non-linear problem is challenging for each coalition, evaluating the Shapley value and the core requires a lot of computational resources for these games. This is an exciting research area of cooperative games, however, with the limitation of the time and experience of the authors, we only focus on the class of cooperative games, where the underlying characteristic functions are described by linear or integer linear programming formulations.

In cooperative games literature, there are some other solution concepts such as the Banzhaf index, the kernel, and the nucleolus, which have been investigated to a great extent. Therefore, it would be interesting to assess and compute these payoff allocations, compare with the core/least core and Shapley value, then decide on the most suitable concept for each operational research game based on the requirements of the players. One can recognise that the properties of fairness and stability are not frequently happening at the same time, except the case when the Shapley value belongs to the core of the game. Therefore, finding the best payoff allocation of an operational research game is a challenging task for the coalitional group in many situations.

We have tested the potential of using the Graver bases method in integer optimisation games. However, it was not as competitive as the algebraic Gröbner method because of the size and the computational time. Both the sizes of Graver bases and Gröbner bases grow exponentially when the dimension of matrix $\mathcal{A}$ and the magnitude of its entries increase. Therefore, the test set methods (Gröbner bases and Graver bases) can only be applied to find the Shapley values for the integer optimisation games with the underlying matrices $\mathcal{A}$ of reasonable sizes.

5.3 Further Research Directions

Solution concepts for operational research games

In cooperative game model, different solution concepts emphasise the separate axioms depending on the group's preference. For example, the core and least-core promote the stability of the game, and the Shapely value focuses on the fairness property. For some versions of Operational Research Games, such as the GMSTG and LPG, it would be interesting if we can also compare the Shapley value and other existing solutions in the Core. In the LPG, the core always exists, and we can compute directly one of its solutions by solving a corresponding linear program. If we can find the Shapley

value exactly or approximately it within ϵ- error, calculating the distance of the Shapley value and the core is appealing to understand the relationship between the core and the Shapley value. On the other hand, one might find a method to combine these sets of properties and build a new solution concept when the Shapley value do not stay in the core.

Other approaches to find an allocation in the core of GMSTG

In the MSTG with the non-empty core, there are some proposed heuristics to find a solution in the core. The similar question is that whether there is a good heuristic to determine a cost allocation in the least core (or the core if it exists) of the GMSTG. Another possible improvement of the constraint generation algorithm is related to the current approach for solving the separation problem, where we utilised CPLEX to resolving the large-scale mixed integer linear program for generating the new cutting plane for each iteration. If the Lagrangian relaxation and sub-gradient method are able to address this separation step, and it might decrease the computational time.

Coalition formulation problem

In the operational research games, we supposed that all the players are willing to collaborate in a grand coalition to share their business resources and agree on the final payoff. This condition is guaranteed by the super-additive property of the characteristic function. In practice, the property is not relevant for some class of operational research games, therefore, the players might want to break out of the grand coalition and build their subgroups. For example, an Integer Optimisation games might not be a super-additive game if the vector $\boldsymbol{d}$ in its formulation has both positive and negative entries. Hence, we have to investigate both the coalition formation structure and the payoff allocation problems in a new perspective for these operational research games.

Potential applications of the Gröbner bases and Graver bases

In general, the sizes and computational times of both Gröbner bases and Graver bases are large for most Integer Programming problems. Therefore, we might concentrate on several classes of Operational Research Games, where one can compute the closed-form formulas of the bases directly. In particular, there are cases where the matrices $\mathcal{A}$ have special structures, the Gröbner bases and Graver bases can then be computed in polynomial time w.r.t. the input parameters. For example, there is a polynomial time algorithm computes the Graver basis of the underlying matrix $\mathcal{A}$ for the N-fold integer program, and the linear integer optimization problem can be solved in polynomial time. In these situations, the test set approach could be very competitive to the other methods for solving the Integer Programs.

Appendix A

Review of the Background

A.1 Cooperative Game Theory

The goal of this part is to reintroduce some basic notation, concepts and the mathematical model of cooperative game theory. We refer the reader to the book by Curiel (1997) for more detailed exposition on the topic of Cooperative Game theory.

A.1.1 Basic concepts

Suppose we have a finite set $N = \{1, 2, \ldots, n\}$ of players, where $n \geq 2$. A coalition is a subset S of the players, where players can coordinate their strategies. In this n player cooperative game, let us denote $2^N \equiv \{S : S \subseteq N\}$ the set of all possible coalitions $S^1, S^2, \ldots, S^{2^n}$. By convention, we call the empty set, $\emptyset$, as the empty coalition and the set N of all players as the grand coalition. In our cooperative game models, it is supposed that all coalitions in the set 2^N are feasible.

In some cooperative games, the set of possible coalitions to a family F can be restricted such as $F \subsetneq \{S^1, S^2, \ldots, S^{2^n}\}$, to model cases where not all coalitions may be feasible in practice. These could be cases when feasible coalitions are the connected subgraphs of a communication graph (Myerson 1977) or when a hierarchy exists on the set of players (Faigle & Kern 1992), we refer the readers to Grabisch (2013) for some further results for this direction.

Definition A.1. A *cooperative game in characteristic function form* G is given by a pair (N, ν) where $N = \{1, 2, \ldots, n\}$ is a finite, non-empty set of players and $\nu : 2^N \to \mathbb{R}$ is a *characteristic function*, which maps each coalition $S \subseteq N$ to a real number $\nu(S)$.

Note that depending on the type of game, $\nu(S)$ is the payoff that coalition S can gain in a value generating game or $c(S)$ is the cost of the group S have to pay if this is a cost paying game. In this section, we review the definitions and theorems in value games and the reader can infer the similar results for the cost games. Most coalition games assume that the condition of empty set has value zero, i.e, $\nu(\emptyset) = 0$. The number $\nu(S)$ may be considered as the value, worth, or power of the coalition S when its members act together.

For any game $G = (S, \nu)$, let $S \subsetneq N$ be a non-empty coalition of players. A $|S|$-player *subgame* $G^S = (S, \nu^S)$ is defined by restricting the game G to coalition contained in S. In particular, its characteristic function $\nu^S : 2^S \to \mathbb{R}$ is naturally defined as $\nu^S(T) := \nu(T)$, for all coalitions $T \subseteq S$.

A.1.2 Game outcome

The next important concept is the outcome of a cooperative game. That must include two parts: a partition of players into coalitions (coalition structure) and a payoff vector, which shares the value of each coalition among its members.

Definition A.2. Given a cooperative game in characteristic function form $G = (N, \nu)$, a *coalition structure* over $N = \{1, \ldots, n\}$ is a collection of non-empty subsets $CS = \{S^1, \ldots S^k\}$ such that $\bigcup_{j=1}^{k} S^j = N$ (each player must appear in some coalition) and $S^i \cap S^j = \emptyset$ for any $i, j \in \{1, \ldots, k\}$ such that $i \neq j$ (a player does not appear in more than one coalition). In this case, each coalition S^j in the coalition structure CS will act independently and achieve its own payoff $\nu(S^j)$.

We now introduce briefly the coalition structure formation problem. When the players come together and form groups, the grand coalition is not the only possible solution. The group-forming step could be done either by the players deciding autonomously using some bargaining procedure or by a system designer. We focus on the second case, where the players are willing to follow the arrangement of a system designer. This designer will consider the social welfare of the system which is higher than the performance of individual players. In the case that the underlying game is a super-additive game, which is correct for most of the games in this thesis, there always exists incentives for the players to work together as a grand coalition and generate the most value.

Definition A.3. A *payoff vector* for a coalition structure $CS = \{S^1, \ldots S^k\}$ over $N = \{1, \ldots, n\}$ is denoted $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ if $x_i \geq 0$ for all $i \in N$ (every player receives a non-negative payoff), and $\sum_{i \in S^j} x_i \leq \nu(S^j)$ for any $j \in \{1, \ldots, k\}$ (the feasibility

requirement of the solution states that the total amount paid out to a coalition does not exceed the total value of that coalition).

Definition A.4. A payoff vector $\boldsymbol{x}$ for a coalition structure CS is said to be an *imputation* if $\sum_{i \in S^j} x_i = \nu(S^j)$ any $j \in \{1, \ldots, k\}$ (efficient condition) and $x_i \geq \nu(\{i\}) \forall i \in N$ (individual rationality condition).

Definition A.5. An *outcome* of cooperative games G is defined as a pair $(CS, \boldsymbol{x})$, where CS is a coalition structure over G and $\boldsymbol{x}$ is a payoff vector for that CS.

A.1.3 Subclasses of cooperative games in characteristic function form

We will now define four subclasses of coalition games: monotone games, simple games, super-additive games and convex games.

- A cooperative game in characteristic function form $G = (N, \nu)$ is said to be *monotone* if it is satisfies $\nu(S^1) \leq \nu(S^2)$ for every pair of coalitions $S^1, S^2 \subseteq N$ such that $S^1 \subseteq S^2$. *Monotone games* have the property that adding a member to any existing coalition cannot decrease the total productivity of this coalition.

- A game $G = (N, \nu)$ is said to be *simple* if it is monotone and its characteristic function only takes the value either 0 and 1, i.e., $\nu(C) \in \{0, 1\}$ for any $C \subset N$. For instance, all coalitions are either “winning" or “losing" in a political game. In a simple game, coalitions of value 1 are said to be winning, and coalitions of value 0 are said to be losing. Such games can model situations where there is a task to be completed: a coalition is winning if and only if it completes the task.

- In *super-additive* games, it is always profitable for two groups of players to join forces. More formally, a cooperative game in characteristic function form $G = (N, \nu)$ is said to be *super-additive* if it satisfies $\nu(S^1 \cup S^2) \geq \nu(S^1) + \nu(S^2)$ for every pair of disjoint coalitions $S^1, S^2 \subseteq N$. Because we have assumed that the value of each coalition is non-negative, super-additivity implies monotonicity.

- A characteristic function ν is said to be *super-modular* if it satisfies $\nu(S^1 \cup S^2) + \nu(S^1 \cap S^2) \geq \nu(S^1) + \nu(S^2)$ for every pair of coalitions $S^1, S^2 \subseteq N$. A game with a super-modular characteristic function is said to be *convex*. In a convex game, we have an intuitive characterization in terms of players marginal contributions: a player is more useful when he joins a bigger coalition.

- A game $G = (N, \nu)$ is *additive* if for all $S^1, S^2 \subset N$, and $S^1 \cap S^2 = \emptyset$, then $\nu(S^1 \cup S^2) = \nu(S^1) + \nu(S^2)$. Note that every additive game is necessarily convex, and all convex games are super-additive games.

Note that in most cooperative games in the thesis, we only deal with the TU games with super-additive property. Hence, the coalition structure is always the grand coalition, i.e. $CS = \{N\}$. Afterwards, we focus only on computing the payoff vector $\mathbf{x}$, which allocates the total payoff $\nu(N)$ to all the players.

A.1.4 Profit and Cost Sharing

From the previous section, we can see the cooperative game model in characteristic function form does not dictate how the coalition value $\nu(S)$ should be divided amongst the members of S. In fact, the question of how to divide coalition value is a fundamental research topic in cooperative game theory. There are a few answers to the question, in the form of solution concepts such as the Shapley value (Shapley 1953), Banzhaf index (Banzhaf 1965), core (Gillies 1959), least core (Maschler et al. 1979), and the nucleolus (Schmeidler 1969), the kernel (David & Maschler 1965). An implicit assumption of cooperative game in characteristic function form is that the coalition value $\nu(S)$ can be divided amongst the members of S in any way that the members of S choose. Games with that property are said to be with transferable utility (TU games).

The Shapley value, the core, least-core have been studied extensively and they are described and surveyed in many papers. We mentioned the formal definition of these concepts as follows: the core and least core in the section 2.2.2 and the Shapley value in the section 3.2.2. Finding the allocations described by these solution concepts is computationally intractable in general. Hence, much of the research in this area focuses on a variety of restricted domains in which one can hope to find either an exact or approximate solution efficiently.

A cooperative game $G = (N, \nu)$ is called *balanced* if and only if the core of the game is non-empty. Moreover, a game $G = (N, \nu)$ is called *totally balanced* if the core of all its subgame $G^S := (S, \nu^S)$ is non-empty for all $S \subseteq N$. Note that the convex game is a totally balanced games since all its core and the cores of any of its subgame are non-empty followed by the Bondareva–Shapley theorem. This theorem was formulated independently by Olga Bondareva (Bondareva 1963) and Lloyd Shapley (Shapley 1967).

A.2 Monte Carlo methods for approximation

This part concentrates on applying the Monte Carlo simulation method to approximate the Shapley value. Mann & Shapley (1960) is the first paper using Monte Carlo simulation to approximate this power index, and then analyse the electoral-vote system in

the US Presidential elections. Recently, there are several other works on using different randomised methods to approximate the Shapley value such as (Michalak et al. 2013, Fatima et al. 2008, Castro et al. 2009).

We now survey the simple random sampling and stratified sampling techniques in Monte Carlo simulation(without presenting all detail). Most of the materials in this part and further exposition on Monte Carlo sampling techniques can be found in the books by Lohr (2010) and Cochran (2007).

A.2.1 Simple Random sampling

We consider the discrete random variable X which has the probability density function $f_X(x)$ as a discrete uniform distribution. Let $g(X)$ is function of random variable X, then the expected value of a function $g(X)$ is of the following form

$$\mathbb{E}(g(X)) = \sum_{x \in \mathrm{X}} g(x) f_X(x).$$

If we can take p samples of X's, which are $\{x_1, \ldots, x_p\}$ and computed the mean of $g(x)$ over the sample, then we would have the Monte Carlo estimate:

$$\overline{g}_p(x) = \frac{1}{p} \sum_{i=1}^{p} g(x_i)$$

of $\mathbb{E}(g(X))$. And the random variable $\overline{g}_p(X) = \dfrac{1}{p} \displaystyle\sum_{i=1}^{p} g(X)$ is called the Monte Carlo estimator of $\mathbb{E}(g(X))$.

If $\mathbb{E}(g(X))$ exists, then the law of large number tell us that for any arbitrarily small ϵ

$$\lim_{p \to \infty} P(|\overline{g}_p(X) - \mathbb{E}(g(X))| \geq \epsilon) = 0.$$

This tells us that as p gets large, then there is small probability that $\overline{g}_p(X)$ deviates much from $\mathbb{E}(g(X))$. Therefore as long as p is large enough, $\overline{g}_p(x)$ arising from a Monte Carlo experiment shall be close to $\mathbb{E}(g(X))$, as desired. Another important thing to note is that $\overline{g}_p(X)$ is unbiased for $\mathbb{E}(g(X))$:

$$\mathbb{E}(\overline{g}_p(X)) = \mathbb{E}(\frac{1}{p} \sum_{i=1}^{p} g(X_i)) = \frac{1}{p} \sum_{i=1}^{p} \mathbb{E}(g(X_i)) = \mathbb{E}(g(X)).$$

To estimate the standard deviation σ of the random variable $g_p(X)$, we use the same p samples as in the case of computing the expectation. A standard estimator for the variance has the following form

$$\mathrm{Var}(g(X)) = \frac{1}{p-1} \sum_{i=1}^{p} (g(X_i) - \mathbb{E}(\overline{g}_p(X)))^2.$$

The corresponding unbiased estimate of the variance of $\overline{g_p}(X)$ is

$$\mathrm{Var}(\overline{g_p}(X)) = \frac{1}{p(p-1)} \sum_{i=1}^{p} (g(X_i) - \mathbb{E}(\overline{g}_p(X)))^2.$$

A.2.2 Stratified sampling techniques

Suppose that the total population M can be divided into distinct subgroups called strata, the stratified sampling technique randomly selects the samples proportionally from the different strata. For each player i in the cooperative game, we have a total budget of m samples to be allocated into n strata of that player. Let us denoted the stratum as $k := \{0, \ldots, n-1\}$, where each stratum contains similar sampling units. If the stratum $\{k\}$ has exactly M_k in the population, we will take m_i^k samples from this stratum, hence, $\sum_{k=0}^{n-1} m_i^k = m$ and $\sum_{k=0}^{n-1} M_k = M$.

Using simple random sampling, we could estimate the stratum mean μ_k based on sample size m_k and sample average $\overline{X_i^k}$. Assume that the number of samples taken from each stratum is sufficiently large so that we can use the central limit theorem to approximate the distribution of X_i^k by a Gaussian with mean $\mu_k = \mathbb{E}(X_i^k)$ and variance $\sigma_{i,k}^2$. Therefore, the vectors $\mu = \{\mu_k\}_{k=0}^{n-1}$ is the vector of strata means and $\sigma = \{\sigma_k\}_{k=0}^{n-1}$ is the vector of strata standard deviation for the stratum.

Stratified sampling offers several advantages over simple random sampling as the stratified sample can provide greater precision than a simple random sample of the same size. We can also apply different sampling methods for different strata depending on the structure of each stratum. We, however, must ensure that sufficient sample points is obtained to support a separate analysis of any stratum. By sampling appropriately from each stratum, the variance of the Shapley value can be smaller compared to the simple random technique.

A major question with stratified surveys is the allocation of observations among the strata. There are number of ways of allocating a sample of size n among the various strata, this leads to three versions of stratified sampling: *proportional allocation sampling, Neyman allocation and optimal allocation sampling.* When the unit sampling costs

of sampling in all strata are similar, which is our case for the Shapley value, the optimal allocation method is the same as the Neyman allocation sampling. We will provide the definition, formal results on the expectations and variances for the two versions.

Proportional allocation. Under proportional allocation, sample sizes are allocated to be proportional to the number of sampling units in the strata. i.e,

$$\frac{m_i^0}{M_0} = \frac{m_i^1}{M_1} = \ldots = \frac{m_i^{n-1}}{M_{n-1}} = \frac{m}{M}, \tag{A.1}$$

In the definition of strata, we have the sizes of n strata are equal, then the similar number of samples can be taken for each strata. For the first and the last strata, we have only one member of each stratum, therefore finding the exact value of this element will bring better precious and time could be safe to sample other strata.

Neyman Allocation. When evaluating the marginal contribution of a player for a coalition requires similar cost, this special case of optimal allocation is known as Neyman allocation (Neyman 1934). The allocation rule is that sample sizes should be proportional to the product of stratum size and the stratum standard deviation in that stratum. Formally, the number of samples taken in the stratum k is shown as follows:

$$m_i^k = \frac{M_k \sigma_k}{\sum_{k=0}^{n-1} M_k \sigma_k} \times m. \tag{A.2}$$

Therefore, large samples are selected for the strata that are larger, more variable, or cheaper to sample in the Neyman allocation. The equation A.2 contains the variance σ_k of stratum k. Hence, it must be computed in advanced for the Neyman sampling; this is normally impractical for a large population. Another approach to allocate the sampling units for the stratum k is by reinforcement learning algorithm. In this approach, depending on the current sets of $\{\sigma_k\}_{k=0}^{n-1}$, the new sampling unit will be chosen in which stratum.

In practice, significant gains in precision are often gained from moving from a random sampling technique to a proportional allocation, but the gain in moving from proportion to Neyman allocation are smaller. Moreover, the Neyman allocation needs the resources to learn the strata' variance in the process. In our algorithm, we selected the proportional stratified sampling to allocate the number of sampling units for each stratum. The chosen unit will affect the variance of that stratum, and the process start again for the next iteration.

A.2.3 LPSA randomised approach

In this part, we describe the details of our proportional stratified sampling in our LPSA randomised algorithm. From the formulation (3.3) of the Shapley value, the stratified random sampling has the following properties: The population consists of $M = n!$ sampling units, which have the form of $S \subset N \setminus \{i\}$, are divided into n groups, called strata. Each stratum k contains the collection of $Q_k = \binom{n-1}{k}$ different coalitions S of the same size $|S| = k$, and $|S|!(n - |S| - 1)!$ copies for each coalition. Hence, there are total $M_k = (n-1)!$ sampling units in each stratum.

For sample allocation among strata, we suppose to take the similar number of observations for each stratum following the equation (A.1) of proportional allocation. However, for the strata k such that $|Q_k| \leq \lfloor m/n \rfloor$, we have only a few distinguish members in each stratum (see, the equation 3.8). Therefore, finding the exact value of these samples in these strata can save the resources to sample other strata.

From each stratum, we then apply the simple random method to approximate the expected value $\mathbb{E}(X_i^k) = \frac{1}{\binom{n-1}{k}} \sum_{|S|=k,\ i \notin S} (\nu(S \cup \{i\}) - \nu(S))$. It is important to realize that the sampling is independent in the different strata. A list of m_i^k samples $\{x_i^{1,k}, \ldots, x_i^{m_k^i,k}\}$ are drawn randomly from each stratum. Therefore the random variable defined by

$$\overline{\phi_i} := \frac{1}{n} \sum_{k=0}^{n-1} \frac{1}{m_i^k} \sum_{j=1}^{m_i^k} x_i^{j,k} = \frac{1}{n} \sum_{k=0}^{n-1} \overline{\mu_i^k},$$

where $\overline{\mu_i^k}$ is the sample mean of the data drawn from stratum k.

The stratified sampling method has proved to attain a better variance than the simple random sampling theoretically. The total variance is the sum of variances from within strata and the variance between strata. Therefore, in general when the variance of between strata is large, simple random sampling is worse than the stratified sampling technique.

Notice that after taking a sample of coalition S, we select all player $\{i\}$ not in S to make a bigger coalition. This process saves the effort to recalculate $\nu(S)$ and have the sensitivity analysis applied $n - |S|$ times to compute the $\nu(S \cup \{i\})$. In particular, instead of finding both $\nu(S \cup \{i\})$ and $\nu(S)$ directly, the Linear Programming sensitivity analysis was utilised to speed up the process of finding the marginal contribution of the player $\{i\}$ to coalition S for large number of recalculations.

A.3 Algebraic Gröbner method for Integer Programs

This section will introduce the algebraic method to solve Integer Linear Program (ILP) where we need some definitions and notation as follows: Gröbner basis, toric ideal of ILP problems, test set, and Graver basis. We initially define the definition of Gröbner bases for an ideal in the ring of polynomials. The Buchberger's algorithm to find such bases is briefly reviewed with more details in the Section 4.2. Afterwards, we show the connection between the Gröbner basis and its test set. An illustrative example of the test set is given and we later show the augmentation step to solve the ILP problems. More detail on Gröbner bases theory and its applications can be found in the works of Buchberger (1985) and Cox et al. (1998).

A.3.1 Definitions and notations

Polynomial ring and term orders

Let denote $\mathbb{R}[\boldsymbol{x}] = \mathbb{R}[x_1, \ldots, x_n]$ the ring of polynomial with coefficients in a field $\mathbb{R}$. The ideal generated by a subset $\mathcal{F} \subset \mathbb{R}[\boldsymbol{x}]$ is the set $< \mathcal{F} >$ consisting of all linear combinations:

$$< \mathcal{F} >= \{h_1 f_1 + \ldots + h_r f_r : f_1, \ldots, f_r \in \mathcal{F}; h_1, \ldots h_r \in \mathbb{R}[\boldsymbol{x}]\}$$

A term order on $\mathbb{Z}_+^n$ is a total order $\prec$ satisfying the following properties:

- $\prec$ is compatible with sums, i.e. $\alpha \prec \beta \Rightarrow \alpha + \gamma \prec \beta + \gamma$, for all $\alpha, \beta, \gamma \in \mathbb{Z}_+^n$.
- $\prec$ is a well-ordering, i.e. $0 \prec \alpha$ for all $\alpha \in \mathbb{Z}_+^n, \alpha \neq 0$.

A lexicographic order $\prec_{lex}$ is a term order where it compares exponents of x_1 in the monomials at the beginning, and in case of equality compares exponents of x_2, and so forth. Given a vector $\boldsymbol{c} \in \mathbb{Z}_+^n$ and two vector $\alpha, \beta \in \mathbb{Z}_+^n$, we say that $\alpha \prec_{\boldsymbol{c}} \beta$ if $\boldsymbol{c}^t\alpha < \boldsymbol{c}^t\beta$ or $\boldsymbol{c}^t\alpha = \boldsymbol{c}^t\beta$, and $\alpha \prec_{lex} \beta$.

For each term order $\prec$, then every non-zero polynomial f has a unique initial term $in_\prec(f) = a\boldsymbol{x}^\alpha$. Let $f = \sum_\alpha a_\alpha \boldsymbol{x}^\alpha$ be a nonzero polynomial in $k[x_1, \ldots, x_n]$ and let $\prec$ be a term order.

- The multidegree of f is defined as: $multideg(f) = \max_\prec\{\alpha \in \mathbb{Z}_+^n \text{ s.t } a_\alpha \neq 0\}$.
- The leading term of f is defined as: $LT(f) = a_{multideg(f)}\ \boldsymbol{x}^{multideg(f)}$.

Gröbner bases and Buchberger's algorithm

Theorem A.6. *(Hilbert's basic theorem.) Every ideal $I \subseteq \mathbb{R}[x_1, \ldots, x_n]$ has a finite generating set. That is $I =< g_1, \ldots, g_s >$ for some $g_1, \ldots, g_s \in I$.*

The interested reader can find a proof of this theorem in the book by Cox et al. (1998). Since the existence of the finite generating set for the monomial ideal $< LT(I) >:=< LT(f) : \forall f \in I >$, we can define the Gröbner basis as follows:

Definition A.7. Given a fixed term order $\prec$, a subset $\{g_1, \ldots, g_s\}$ of an ideal I is called a *Gröbner basis* if $< LT(I) >=< LT(g_1), \ldots, LT(g_s) >$. Equivalently, $\{g_1, \ldots, g_s\}$ is a Gröbner basis of ideal I if and only if the leading term of any element in I is divisible by one of the leading term $LT(g_i)$, $i \in \{1, \ldots, s\}$.

In computational algebraic geometry, the Buchberger's algorithm is the method to transform a set of generators for a polynomial ideal into a Gröbner basis with respect to a term order $\prec$. One can view it as a generalization of the Euclidean algorithm for univariate GCD computation and of Gaussian elimination for linear systems (Sturmfels 1996). The algorithm can be improved by a geometric version of Buchberger's algorithm, which exploits the special structure of toric ideals in order to achieve a better performance (Thomas 1995).

The next part investigates the Gröbner basis for a toric ideal and introduces the test set for integer program. Moreover, there is a bijective function which transforms the Gröbner basis to the test set and an illustrative example will be shown.

A.3.2 Test sets for the Integer Linear Program

Integer Program Model and its Toric Ideal

Consider an integer linear programming problem $ILP_{(\mathcal{A},\boldsymbol{c})}(\boldsymbol{b})$ as follows:

$$\begin{aligned} ILP_{(\mathcal{A},\boldsymbol{c})}(\boldsymbol{b}): \quad \max \ f(\boldsymbol{x}) &= \boldsymbol{c}^t\boldsymbol{x} \\ \text{s.t.} \quad \mathcal{A}\boldsymbol{x} &= \boldsymbol{b}, \\ \boldsymbol{x} &\in \mathbb{Z}^n_+, \end{aligned} \tag{A.3}$$

where $\mathcal{A} \in \mathbb{Z}^{m\times n}$, $\boldsymbol{b} \in \mathbb{Z}^m$, $\boldsymbol{c} \in \mathbb{R}^n$. The notation $ILP_{(\mathcal{A},\boldsymbol{c})}(b)$ denotes the integer linear programming problem with the fixed matrix $\mathcal{A}$, cost vector $\boldsymbol{c}$, and the right hand side $\boldsymbol{b}$. The notation $ILP_{(\mathcal{A},\boldsymbol{c})}$ denotes the set of all the integer linear programming problems obtained by varying the right hand side vector $\boldsymbol{b}$, fixed $\mathcal{A}$ and the cost function $\boldsymbol{c}$.

We define π by the map $\pi(\boldsymbol{x}) = \mathcal{A}\boldsymbol{x}$ for $\boldsymbol{x} \in \mathbb{Z}_+^n$. Given a vector $\boldsymbol{b} \in \mathbb{Z}^d$, the set $\pi^{-1}(\boldsymbol{b}) = \{u \in \mathbb{Z}_+^n : \pi(u) = \boldsymbol{b}\}$ is called the $\boldsymbol{b}$-fibre of $ILP_{(\mathcal{A},\boldsymbol{c})}$.

With the cost vector $\boldsymbol{c}$, we fixed an associated term order $\prec_{\boldsymbol{c}}$ as defined in previous part. Assumed that for each rhs vector $\boldsymbol{b}$ there exists an optimum solution for $ILP_{(\mathcal{A},\boldsymbol{c})}(\boldsymbol{b})$. We define a set $G_{\mathcal{A},\prec_{\boldsymbol{c}}} \subset \mathbb{Z}^n$ is a *test set* for the family of integer linear problems w.r.t. the matrix $\mathcal{A}$ and the order $\prec_{\boldsymbol{c}}$ if:

- for each non-optimal point α of $ILP_{(\mathcal{A},\boldsymbol{c})}(\boldsymbol{b})$, there exists $g \in G_{\mathcal{A},\prec_{\boldsymbol{c}}}$ such that $\alpha - g$ is a feasible solution for $ILP_{(\mathcal{A},\boldsymbol{c})}(\boldsymbol{b})$ and $\alpha - g \prec_{\boldsymbol{c}} \alpha$,
- for the optimal point β, $\beta - g$ is not a feasible point for any $g \in G_{\mathcal{A},\prec_{\boldsymbol{c}}}$.

To find out the Gröbner test set for an ILP problem, we need to depend on the definition of toric ideal, which comes from the field of Algebraic Geometry. Formally, given a matrix $\mathcal{A}$, the *toric ideal* associated with this matrix is denoted as $I_{\mathcal{A}}$ as following:

$$I_{\mathcal{A}} =< \boldsymbol{x}^{\alpha} - \boldsymbol{x}^{\beta} \text{ s.t. } \alpha - \beta \in Ker(\mathcal{A}) \ \& \ \alpha, \beta \in \mathbb{Z}_+^n > \ \subset \mathbb{Q}[x_1, \ldots, x_n].$$

By calculating the Gröbner basis of the toric ideal of matrix $\mathcal{A}$, we can transform this basis to the required test set for the $ILP_{(\mathcal{A},\boldsymbol{c})}(\boldsymbol{b})$ problem. In the next step, we construct a minimal basis of this ideal called the reduced Gröbner basis with some special properties. The most important connection of the previous defined terms is shown in the next results.

Gröbner bases of the toric ideals in Integer Linear Program

The relationship between the previous concepts is that the reduced Gröbner basis $\mathcal{G}_{\mathcal{A},\prec_{\boldsymbol{c}}}$ of $I_{\mathcal{A}}$ with respect to the order $\prec_{\boldsymbol{c}}$ allows us to compute a uniquely defined minimal test set $G_{\mathcal{A},\prec_{\boldsymbol{c}}}$ for the ILP (A.3). The reduced Gröbner basis is formed by binomials

$$\mathcal{G}_{\mathcal{A},\prec_{\boldsymbol{c}}} = \{\boldsymbol{x}^{\alpha_i} - \boldsymbol{x}^{\beta_i}, i = 1, 2, \ldots, r\}, \text{ with } in_{\prec_{\boldsymbol{c}}}(\boldsymbol{x}^{\alpha_i} - \boldsymbol{x}^{\beta_i}) = \boldsymbol{x}^{\alpha_i}.$$

From the reduced Gröbner basis, the test set is expressed as:

$$G_{\mathcal{A},\prec_{\boldsymbol{c}}} = \{\alpha_i - \beta_i; \text{ for } i = 1, 2, \ldots, r\}$$

Theorem A.8. *(Conti & Traverso 1991) For each cost vector $\mathbf{c}$ and a matrix $\mathcal{A}$, a reduced Gröbner basis $\mathcal{G}_{\mathcal{A},\prec_{\boldsymbol{c}}}$ of $I_{\mathcal{A}}$ generates a test set $G_{\mathcal{A},\prec_{\boldsymbol{c}}}$ for the family of $ILP_{(\mathcal{A},\boldsymbol{c})}$.*

The test set $G_{\mathcal{A},\prec_{\boldsymbol{c}}}$ connects the fibre $\pi^{-1}(\boldsymbol{b})$ as a directed graph where a unique sink is an optimal solution. After acquiring the test set, we implement a search algorithm called *augmentation algorithm* to solve the integer linear program for each rhs vector $\boldsymbol{b}(S)$, starting from a feasible solution of the problem. At every step of the augmentation algorithm, we have two different cases:

- There exists an element in the test set which, when subtracted from the current point, yields an improved point.
- There does not exist such an element in the set so that the point is the optimum of the fibre.

Graver test set for Integer Linear Programs

Another popular test set for the family of integer programs $ILP_{(\mathcal{A},\boldsymbol{c})}(\boldsymbol{b},\boldsymbol{c})$ is called Graver test set (Graver 1975), and denoted as $Gr_{\mathcal{A}}$ when varing the rhs $\boldsymbol{b}$ and cost vector $\boldsymbol{c}$ in the problem (A.3). This test set is closely related to the Hilbert bases of polyhedral cone. Formally, the Graver basis of the integer matrix $\mathcal{A}$ is the finite set $Gr_{\mathcal{A}}$ of minimal elements in the set

$$\{\boldsymbol{x} \in \mathbb{Z}^n : \mathcal{A}\boldsymbol{x} = 0, \boldsymbol{x} \neq 0\}$$

w.r.t. a partial order on $\mathbb{Z}^n$ defined by $x \sqsubseteq y$ when $x_i y_i \geq 0$ and $|x_i| \leq |y_i|$ for all i.

Moreover, the Graver basis contains all the Gröbner bases w.r.t any term order $\prec_{\boldsymbol{c}}$ (Sturmfels 1996). In particular, $\bigcup_{\boldsymbol{c}} G_{\mathcal{A},\prec_{\boldsymbol{c}}} \subset Gr_{\mathcal{A}}$. This shows that the size of the Gröbner basis is normally less than the size of Graver basis. We refer the reader to the book by Onn (2010) for more detail exposition on the algorithmic theory of Graver bases and its application to integer linear (or non-linear) programming. An example will be provided to show the Gröbner basis and Graver basis of a simple Integer Program.

Example A.1. *Given a matrix*

$$\mathcal{A} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \end{pmatrix}.$$

and a vector $\boldsymbol{c} = \{1, 3, 5, 7\}$*, we build an integer program (A.3) with varied rhs* $\boldsymbol{b}$*. The corresponding map* $\pi : \mathbb{R}^4 \to \mathbb{R}^2$ *is defined by* $\pi(\boldsymbol{x}) = \mathcal{A}\boldsymbol{x}$*.*

Hence, the toric ideal $I_{\mathcal{A}} =< x_1x_3 - x_2^2, x_2x_4 - x_3^2, x_1x_4 - x_2x_3, x_1^2x_4 - x_2^3, x_1x_4^2 - x_3^3 >$*. Finding the reduced Gröbner basis of this toric ideal* $I_{\mathcal{A}}$ *w.r.t. the term order* $\prec_c$*, we can build a test set as following:*

$$\pm(1, -2, 1, 0); \pm(0, 1, -2, 1); \pm(1, -1, -1, 1)$$

However, the Graver basis will be as follows:

$$\pm(1,-2,1,0);\pm(0,1,-2,1);\pm(1,-1,-1,1);\pm(2,-3,0,1);\pm(1,0,-3,2)$$

After we find a test set T for $ILP_{(\mathcal{A},\boldsymbol{c})}(\boldsymbol{b})$ (such as the Gröbner test set $G_{\mathcal{A},\prec_{\boldsymbol{c}}}$ or Graver test set $Gr_{\mathcal{A}}$) and a feasible solution z_0 to the ILP problem are available, the following augmentation algorithm can be employed in order to solve the integer programming problem.

A.3.3 Augmentation algorithm in the Algebraic Gröbner approach

Augmentation algorithm is similar to the simplex method in the sense that the objective values improve through each iteration of the algorithms. Moreover, we can also apply the stage-one of the augmentation algorithm to find a feasible solution for the integer programming problem.

Algorithm 6 (Augmentation Algorithm with the parameter rhs $\boldsymbol{b}$)

Input:
- $\mathcal{A}, \boldsymbol{b}, \boldsymbol{c}$ where $\mathcal{A}$ is the matrix, $\boldsymbol{b}$ is rhs, and the cost vector $\boldsymbol{c}$ in (A.3),
- The test set T for $ILP_{(\mathcal{A},\boldsymbol{c})}(\boldsymbol{b})$ (using 4ti2 software),
- A feasible solution z_0 to $ILP_{(\mathcal{A},\boldsymbol{c})}(\boldsymbol{b})$ (solving a matrix decomposition problem).

Output: an optimum z_{min} of $ILP_{(\mathcal{A},\boldsymbol{c})}(\boldsymbol{b})$

while $\exists t \in T$ & $\lambda \in \mathbb{Z}_{>_0}$ such that $z_0 + \lambda t$ feasible & $f(z_0 + \lambda t) > f(z_0)$ **do**
  Set $z_0 := z_0 + \lambda t$

end while

return $z_{max} = z_0$

In the augmentation algorithm, the process of finding an optimal solution to $ILP_{(\mathcal{A},\boldsymbol{c})}(\boldsymbol{b})$ via test sets can be described as follows: Initially, we locate a feasible solution z_0 of $ILP_{(\mathcal{A},\boldsymbol{c})}(\boldsymbol{b})$. Then it checks whether this solution z_0 is optimal by finding a potential better feasible solution. If there is no such solution, the current one is the optimal solution. Otherwise, it moves to the next solution and repeats the iteration to check the optimality condition.

The algorithm has an assumption that we can use the Gröbner basis to find out the initial feasible solution. This step is a non-trivial problem in general. However, as we have the Gröbner basis, there is a procedure similar to Phase I of the simplex method in linear programming to find such initial feasible solution. First, we allocate an initial solution $z_0 \in \mathbb{Z}^n$ to $\mathcal{A}z = \boldsymbol{b}$ by using Hermite normal form. Then, the algorithm help us move closer to the feasible area by subtracting suitable vector from $G_{\prec_{\boldsymbol{c}}}$ such that

$$\|(z_0 + \lambda t)^-\|_1 < \|(z_0)^-\|_1,$$

where $\|\boldsymbol{x}\|_1 = \sum_{i=1}^{n} |x_i|$ if $\boldsymbol{x} = \{x_i\}_{i=1}^{n}$ and $(\boldsymbol{x})^- := min\{x, 0\}$. Then, we will replace z_0 by $z_0 + \lambda t$ and repeat until a feasible solution is attained.

The step of of finding the next feasible solution requires the checking of all tuples (t, λ) in $(T, \mathbb{Z})$. It depends on the structures of the feasible area and the objective function f that a relevant method can be applied. In Section 4.4.1, when the objective function $f = \boldsymbol{c}^t\boldsymbol{x}$, we describe an heuristic method for this process.

References

Aadithya, K. V., Ravindran, B., Michalak, T. P., & Jennings., N. R. (2010), 'Efficient Computation of the Shapley Value for Centrality in Networks', *Internet Workshop on Internet and Network Economics, Lecture Notes in Computer Science* **6484**, 1–13.

Ando, K. (2012), 'Computation of the Shapley Value of Minimum Cost Spanning Tree Games: ♯P-Hardness and Polynomial Cases', *Japan Journal of Industrial and Applied Mathematics* **29**(3), 385–400.

Aumann, R. J.(2009), 'Some non-superadditive games, and their Shapley values, in the Talmud', *International Journal of Game Theory* **39**(1), 3–10.

Banzhaf, J. F.(1965), 'Weighted voting doesn't work: A mathematical analysis', *Rutgers Law Review* **19**, 317–343.

Bergantiños, G. & Gómez-Rúa, M.(2012), 'An axiomatic approach in minimum cost spanning tree problems with groups', *Annals of Operations Research* **225**, 45–63.

Bertsimas, D. & Demir, R.(2002), 'An Approximate Dynamic Programming Approach to Multidimensional Knapsack Problems', *Management Science* **48**(4), 550–565.

Bertsimas, D., Perakis, G. & Tayur, S.(2000), 'A new algebraic geometry algorithm for Integer programming', *Management Science* **46**(7), 999–1008.

Bertsimas, D. & Tsitsiklis, J. N.(1997), *Introduction to Linear Optimization*. Massachusetts: Athena Scientific Belmont.

Bhagat, S., Kim, A., Muthukrishnan, S. & Weinsberg, U.(2014), 'The Shapley Value in Knapsack Budgeted Games', *Web and Internet Economics, Lecture Notes in Computer Science* **8877**, 106–119.

Bigatti, A., LaScala, R. & Robbiano, L. 'Computing toric ideals'. *Journal of Symbolic Computation* **27**, 351–365.

Bilbao, J. M., Fernandez, J. R., Jimenez, N. & Lopez, J. J.(2002). 'Voting power in the European union enlargement'. *European Journal of Operational Research* **143**, 181–196.

Bird, C.G.(1976), 'On cost allocation for a spanning tree: A game theory approach', *Networks*, **6**, 335–350.

Bjorndal, E. & Jornsten, K.(2009), 'Lower and upper bounds for linear production games', *European Journal of Operational Research*, **196**(2), 476–486.

Bogomolnaia, A. & Moulin, H.(2010), 'Sharing a minimal cost spanning tree: Beyond the Folk solution', *Games and Economic Behaviour* **69**(2), 238–248.

Bondareva, O.N.(1963). 'Some applications of linear programming methods to the theory of cooperative games (In Russian)' (PDF). *Problemy Kybernetiki* **10**, 119–139.

Borm, P., Hamers, H. & Hendrick, R.(2001), 'Operations research games: A survey', *Top*, **9**(2), 139–199.

Buchberger, B.(1985), 'Gröbner bases: An algorithmic method in polynomial ideal theory', *Multidimensional Systems Theory and Applications. Progress, Directions and Open Problems in Multidimensional Systems*, Reidel Publishing Company, Editors: N. K Bose, 184–232.

Buchberger, B.(1965), *An Algorithm for Finding the Basis Elements of the Residue Class Ring of a Zero Dimensional Polynomial Ideal.* Ph.D. dissertation, University of Innsbruck. English translation in *Journal of Symbolic Computation* (2016) **41**(3–4), 471–511.

Braess, D., Nagurney, A. & Wakolbinger, T.(2005), 'On a paradox of traffic planning', *Transportation Science*, **39**(4), 446–450.

Castro, F., Gago, J., Hartillo, I., Puerto, J. & Ucha, J. M.(2011), 'An algebraic approach to integer portfolio problems'. *European Journal of Operations Research* **210**(3), 647–659.

Castro, J., Gómez, D. & Tejada, J. (2009), 'Polynomial calculation of the Shapley value based on sampling'. *Computers & Operations Research*, **36**(5), 1726–1730.

Caprara, A. & Letchford, A. N.(2010), 'New techniques for cost sharing in combinatorial optimization games', *Mathematical Programming, Ser. B* **124**, 93–118.

Chalkiadakis, G., Elkind, E. & Wooldridge, M.(2011), *Computational Aspects of Cooperative Game Theory*, Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan Claypool Publishers.

Chatterjee, K. & Samuelson, W. F. (2001), *Game theory and business applications*. New York. Kluwer Academic Publishers.

Claus, A. & Kleitman, D. J.(1973), 'Cost allocation for a spanning tree', *Networks* **3**, 289–304.

Cochran, W. G.(2007), *Sampling Techniques*. 3rd Wiley Student Edition. New York: John Wiley & Sons.

Conti, P. & Traverso, C.(1991), 'Gröbner bases and integer programming', *Proceeding AAECC-9 (New Orleans), Springer Verlag, Lecture Notes in Computer Science* **539**, 130–139.

Cox, D., Little, J. & O'Shea, D.(1998), *Using Algebraic Geometry*. New York-Berlin-Heidelberg: Springer-Verlag.

Curiel, I.(1997), *Cooperative Game Theory and Applications: Cooperative Games arising from Combinatorial Optimization Problems*. Dordrecht: Kluwer Academic Publishers.

Davis, M. & Maschler, M.(1965), 'The kernel of a cooperative game'. *Naval Research Logistics Quarterly*, **12**, 223–259.

Deng, X. & Papadimitriou, C. H.(1994), 'On the complexity of cooperative solution concepts'. *Mathematics of Operations Research* **19**(2), 257–266.

Deng, X., Ibaraki, T. & Nagamochi, H.(1999), 'Algorithmic Aspects of the Core of Combinatorial Optimization Games'. *Mathematics of Operations Research* **24**(3), 751–766.

Dinar, A., Ratner, A. & Yaron, D.(1992), 'Evaluating Cooperative Game Theory in water resources', *Theory and Decision*, **32**(1), 1–20.

Faigle, U. & Kern, W.(1997), 'The Shapley value for cooperative games under precedence constraints', *International Journal of Game Theory*, **21**(3), 249–266.

Faigle, U. , Kern, W., Fekete, S. P. & Hochstättler, W.(1997), 'On the Complexity of Testing Membership in the Core of Min-Cost Spanning Tree Games', *International Journal of Game Theory* **26**(3), 361–366.

Fatima, S.S., Wooldridge, M. & Jennings, N. R.(2008), 'A linear approximation method for the Shapley value', *Artificial Intelligence* **172**, 1673–1699.

Feremans, C., Labbé, M. & Laporte, G. (2002), 'A comparative analysis of several formulations for the generalised minimum spanning tree problem', *Networks* **39**(1), 29–34.

Fernández, F.R., Fiestras-Janeiro, M.G., García-Jurado, I., & Puerto, J.(2005), 'Competition and Cooperation in Non-Centralized Linear Production Games'. *Annals of Operations Research* **137**(1), 91–100.

Fiestras-Janeiro, M. G., Garcia-Jurado, I., Meca, A. & Mosquera, M. A.(2011), 'Cooperative game theory and inventory management', *European Journal of Operational Research* **210**(3), 459–466.

Frisk, M. , Göthe-Lundgren, M. & Jörnsten, K. & Rönnqvist, M.(2010), 'Cost allocation in collaborative forest transportation', *European Journal of Operational Research* **205**(2), 448–458.

Gago, V., Manuel, J., Hartillo, H., Maria, I., Puerto, A., Justo, U. E. & Jose, M.(2013), 'Exact cost minimization of a series-parallel reliable system with multiple component choices using an algebraic method', *Computers & Operations Research* **40**(11), 2752–2759.

Gago-Vargas J. , Hartillo, I., Puerto, J. & Ucha, J. M. (2015), 'An improved test set approach to nonlinear integer problems with applications to engineering design', *Computational Optimization and Applications* **62**(2), 565–588.

Gerla, M. & Frata, L.(1988), 'Tree structured fiber optics MAN's'. *Selected Areas in Communications IEEE Journal* **6**, 934–943.

Gilles, D. B. (1959), 'Solutions to general non-zero-sum games', *Contributions to the theory of games*. Princeton University Press.

Gilmore, P.C. & Gomory, R. E.(1963), 'A Linear Programming Approach to the Cutting Stock Problem-Part II', *Operations Research* **11**(6), 863–888.

Goemans, M. X. & Skutella, M.(2004), 'Cooperative facility location games'. *Journal of Algorithms* **50**, 194–214.

Golden, B., Raghavan, S. & Stanojević, D.(2005), 'Heuristic Search for the Generalized Minimum Spanning Tree Problem'. *INFORMS Journal on Computing* **17**, 290–304.

Gómezb, D., Aranguenab, E.G., Manuelb, C., Owena, G., Pozob, M. & Tejada, J.(2003), 'Centrality and power in social networks: a game theoretic approach', *Mathematical Social Sciences* **46**(1), 27–54.

Gow, S. H. & Thomas, L.C.(1998), 'Interchange fees for bank ATM networks'. *Naval Research Logistics* **45**(4), 407–417.

Göthe-Lundgren, M., Jörnsten, K. & Värbrand, P.(1996), 'On the nucleolus of the basic vehicle routing game', *Mathematical Programming* **72**, 83–100.

Grabisch, M. (2013), 'The core of games on ordered structures and graphs'. *Annals of Operations Research* **204**, 33–64.

Granot, D. (1986), 'A generalised linear production model: a unifying model', *Mathematical Programming* **34**, 212–222.

Granot D. & Huberman, G.(1981), 'Minimum cost spanning tree games', *Mathematical Programming* **21**, 1–18.

Graver, J. E.(1975), 'On the foundations of linear and integer programming', *Mathematical Programming* **8**, 207–226.

Guardiola, L. A., Meca, A. & Puerto, J.(2009), 'Production-inventory games: A new class of totally balanced combinatorial optimization games', *Games and Economic Behavior* **65**(1), 205–219.

Hemmecke, R. & Malkin, P. N.(2009), 'Computing generating sets of lattice ideals and Markov bases of lattices', *Journal of Symbolic Computation* **44**(10), 1463–1476.

Hemmecke, R. Onn, S. & Weismantel, R.(2011), 'A polynomial oracle-time algorithm for convex integer minimization', *Mathematical Programming, Ser.A* **126**(1), 97–117.

Herrero, M. J.(1989), 'The nash program: Non-convex bargaining problems', *Journal of Economic Theory* **49**(2), 266–277.

Hoeffding, W.(1963), 'Probability inequalities for sums of bounded random variables', *Journal of the American Statistical Association* **58**(301), 13–30.

Horowitz, E., Sahni, S. & Rajasekaran, S. (1978), *Fundamentals of Computer Algorithms*. New York: Computer Science Press.

Hosten, S. & Sturmfels, B. (1995), 'GRIN: An implementation of Gröbner bases for integer programming', In: *Integer programming and combinatorial optimisation, Lecture Notes in Computer Science* **920**, 267–276.

Kahan, J. & Rapoport, A.(1984), *Theories of coalition formation*. London: Lawrence Erlbaum Associates.

Kraus, S.(1997), 'Negotiation and cooperation in multi agent environments', *Artificial Intelligence* **94**(1), 79–97.

Kruskal, J. B.(1956), 'On the shortest spanning tree of a graph and the travelling salesman problem', *Proceedings of the American Mathematical Society* **7**, 48–50.

Le, P. H, Nguyen, T. D, & Bektaş, T.(2016), 'Generalized minimum spanning tree games', *EURO Journal on Computational Optimization* **4**(2), 167–188.

Lemaire, J.(1991), 'Cooperative game theory and its insurance applications', *ASTIN Bulletin* **21**(1), 7–41.

Leng, M. & Parlar, M.(2009), 'Allocation of Cost Savings in a Three-Level Supply Chain with Demand Information Sharing: A Cooperative-Game Approach', *Operations Research* **57**(1),200-213.

Littlechild, S.C. & Thompson, G.F.(1977), 'Aircraft Landing Fees: A Game Theory Approach', *The Bell Journal of Economics* **8**(1), 186–204.

De Loera, J., Hemmecke, R. & Koppe, M.(2013), *Algebraic and Geometric Ideas in the Theory of Discrete Optimization*. Philadelphia, PA: Society for Industrial and Applied Mathematics.

Lohr, S. L.(2010), *Sampling: Design and Analysis*, second edition. Advanced Series. US: Cengage Learning.

Lozano, S., 'DEA production games', *European Journal of Operational Research* **231**(2), 405–413.

Maleki, S., Tran, T.L., Hines, G., Rahwan, T. & Rogers, A.(2013), 'Bounding the Estimation Error of Sampling-based Shapley Value Approximation', *CoopMAS 2014, AAMAS 14*.

Mann, I. & Shapley, L. S.(1960), 'Values for large games, IV: Evaluating the Electoral College by Monte Carlo Techniques' .Santa Monica, CA: RAND Corporation, 1960. <http://www.rand.org/pubs/research_memoranda/RM2651.html>.

Maschler, M., Peleg, B. & Shapley, L. S.(1979), 'Geometric properties of the kernel, nucleolus, and related solution concepts', *Mathematics of Operations Research* **4**, 303–338.

Maschler, M., Solan, E. & Zamir, S.(2013), *Game Theory*. Cambridge: Cambridge University Press.

Mathur, S. , Sankaranarayanan, L. & Mandayam, N. (2008), 'Coalitions in cooperative wireless networks', *IEEE Journal on Selected Areas in Communications* **26**, 1104–1115.

Michalak, T. P., Aadithya, K. V., Szczepanski, P. L., Ravindran, B. & Jennings, N. R.(2013), 'Efficient Computation of the Shapley Value for Game-Theoretic Network Centrality', *Journal of Artificial Intelligence Research* **46**, 607–650.

Molina, E. & Tejada, J.(2004), 'Linear production games with committee control: Limiting behaviour of the core', *European Journal of Operational Research* **154**, 609–625.

Myerson, R. B.(1977), 'Graphs and cooperation in games', *Mathematics of Operations Research* **2**, 225–229.

Myung, Y. S., Lee, C. H. & Tcha, D. W.(1995), 'On the Generalised Minimum Spanning Tree Problem', *Networks* **26**, 231–241.

Nagarajan, M. & Sošić, G.(2008), 'Game-theoretic analysis of cooperation among supply chain agents: Review and extensions', *European Journal of Operational Research* **187**(3), 719–745.

Narayanam, R. & Narahari, Y.(2010), 'A Shapley value-based approach to discover influential nodes in social networks', *IEEE Transactions on Automation Science and Engineering* **8**(1), 130–147.

Nash, J. F.(1950), 'Equilibrium points in N-person games', *Proceedings of the National Academy of Sciences of the United States of America* **36**, 48–49.

Nash, J.F.(1953), 'Two person cooperative games', *Econometrica* **21**, 128–140.

Von Neumann, J. & Morgenstern, O. (1944), *Theory of Games and Economic Behavior*. Princeton: Princeton University Press.

Neyman, J.(1934), 'On the two different aspects of the representative methods. The method stratified sampling and the method of purposive selection', *Journal of Royal Statistical Society* **97**, 558–606.

Nishizaki, I., Hayashida, T., Shintomi, Y.(2016). A core-allocation for a network restricted linear production game. Annals of Operations Research, 238(1), 389–410.

O'Brien, G., Gamal, A.E. & Rajagopal, R.(2015), 'Shapley Value Estimation for Compensation of Participants in Demand Response Programs', *IEEE Transactions on Smart Grid* **6**, 2837–2844.

Onn, S.(2010), *Nonlinear Discrete Optimization. An Algorithmic Theory* . Zurich, Switzerland: European Mathematical Society Publishing House.

Owen, G.(1975), 'On the core of linear production games', *Mathematical Programming* **9**, 358–370.

Peterson, C. C.(1967), 'Computational experience with variants of the Balas algorithm applied to the selection of research and development projects', Management Science **13**, 736-–750.

Pop, P.C. (2009), 'A survey of different integer programming formulations of the generalized minimum spanning tree problem', *Carpathian Journal of Mathematics*, **25**,104 – 118.

Prim, R.C.(1957), 'Shortest connection networks and some generalizations', *Bell System Technical Journal* **36**, 1389–1401.

Prisco, J. J.(1986), 'Fiber optic regional area networks in New York and Dallas', *IEEE Journal in Selected Areas in Communications* **4**, 750–757.

Puchinger, J., Raidl, G. R. & Pferschy, U.(2010), 'The Multidimensional Knapsack Problem: Structure and Algorithms', *INFORMS Journal on Computing* **22**(2),250–265.

Rubinstein, A.(1982), 'Perfect equilibrium in a bargaining model', *Econometrica* **50**(1), 97–109.

Serrano, R.(2005), 'Fifty years of the Nash program', *Investigaciones Económicas* **29**(2), 219–258.

Scarf, H. E.(1981), 'Production sets with indivisibilities, Part I: Generalities', *Econometrica*, 49,1–32.

Schmeidler, D.(1969), 'The nucleolus of a characteristic function game', *SIAM Journal on Applied Mathematics* **17**, 1163–1170.

Shapley, L. S.(1952), 'A value for n-person games', Santa Monica, CA: RAND Corporation. <http://www.rand.org/pubs/papers/P0295.html.

Shapley, L. S. & Shubik, M.(1954), 'A Method for Evaluating the Distribution of Power in a Committee System', *American Political Science Review* **48**, 787–792.

Shapley, L. S. (1967). 'On balanced sets and cores'. *Naval Research Logistics Quarterly* **14**, 453—460.

Shapley, L. S. & Shubik, M.(1971), 'The assignment game I: the core', *International Journal of Game Theory* **1**, 111–130.

Shubik, M.(1962), 'Incentives, decentralized control, the assignment of joint costs and internal pricing', *Management Science* **8**(3), 325–343.

Sturmfels, B. (1996), *Gröbner Bases and Convex Polytopes*. American Mathematical Society. **8**. Providence.

Sturmfels, B. & Thomas, R. R.(1997), 'Variation of cost functions in integer programming', *Mathematical Programming* **77**(2), 357–387.

Tayur, S. R., Thomas, R. R. & Natraj, N. R.(1995), 'An algebraic-geometry algorithm for scheduling in presence of setups and correlated demands', *Mathematical Programming* **69**(3), 369–401.

Tamir, A. (1988), 'On the core of a travelling salesman cost allocation game', *Operations Research Letters* **8**, 31–34.

Tamir, A.(1991), 'On the core of network synthesis games', *Mathematical Programming* **50**(1), 123–135.

Tijs, S., Branzei, R., Moretti, S. & Norde, H.(2006), 'Obligation rules for minimum cost spanning tree situations and their monotonicity properties', *European Journal Operational Research* **175**, 121–134.

Thomas, R. R. (1995), 'A Geometric Buchberger Algorithm for Integer Programming', *Mathematics of Operations Research* **20**(4), 864–884.

Vickrey, W. (1961), 'Counter speculation, auctions, and competitive sealed tenders', *Journal of Finance* **16**, 8–37.

Weingartner, H. M.(1966), 'Capital budgeting of interrelated projects: survey and synthesis', *Management Science* **12**, 485–516.

Weismantel, R.(1998), 'Test sets of integer programs', *Mathematical Methods of Operations Research* **47**(1),1–37.

Wolsey, L. A. & Nemhauser, G. L.(1999), *Integer and Combinatorial Optimization*. USA: John Wiley & Sons.

Wooldridge, M. & Dunne, P. E.(2006), 'On the computational complexity of coalitional resource games', *Artificial Intelligence* **170**, 835–871.

Young, H. P.(1985), 'Monotonic solutions of cooperative games', *International Journal of Game Theory* **14**(2), 65–72.

Zverovich, A. (2002), GTSP instances. www.cs.rhul.ac.uk/home/zvero/GTSPLIB/

IBM ILOG CPLEX, optimization model development toolkit for mathematical and constraint programming. <http://www-03.ibm.com/software/products/en/ibmilogcpleoptistud/>

4ti2 team, 4ti2 a software package for algebraic, geometric and combinatorial problems on linear spaces. <http://www.4ti2.de/>