MULTIPLE CONNECTIVITY APPROACH TO NETWORK FORMATION GAMES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF APPLIED MATHEMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

CAN DENİZ ÇAM

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
SCIENTIFIC COMPUTING

DECEMBER 2023

Approval of the thesis:

MULTIPLE CONNECTIVITY APPROACH TO NETWORK FORMATION GAMES

submitted by **CAN DENİZ ÇAM** in partial fulfillment of the requirements for the degree of **Master of Science in Scientific Computing Department, Middle East Technical University** by,

Name, Last Name:	CAN DENİZ ÇAM
Signature :	

ABSTRACT

MULTIPLE CONNECTIVITY APPROACH TO NETWORK FORMATION GAMES

Çam, Can Deniz
M.S., Department of Scientific Computing
Supervisor : Assoc. Prof. Dr. Esma Gaygısız
Co-Supervisor : Assoc. Prof. Dr. Hamdullah Yücel

December 2023, 116 pages

The analyses of network structures, formations and stability characteristics are gaining importance and attracting increasing attention. This thesis establishes a specific network formation game and suggests a network stability concept. The game is among firms forming networks to reduce their costs and, hence, increase their payoffs. The multiple connectivity approach computes how individuals may alter the combinations of their connections to reach optimal payoff levels by solving a mixed integer optimization problem. Finding the optimal combinations of multiple discrete connection choices creates a complex problem. This problem is solved by implementing an interpolation method that is particular to the problem and modifying the branch and bound algorithm. The ability of the proposed model to describe network formation is investigated by simulating different network structures emanating from the different model parameters. These simulations hint at the existence of multiple connectivity stability for networks, even in the cases where pairwise stability can not be provided within the context of the established model of the thesis. We take this as the indication of a new stability concept: multiple connectivity stability for networks.

Keywords: Network Formation, Network Stability, Mixed Integer Nonlinear Optimization

ÖZ

AĞ OLUŞUMU OYUNLARINA ÇOKLU BAĞLANTI YAKLAŞIMI

Çam, Can Deniz
Yüksek Lisans, Bilimsel Hesaplama Bölümü
Tez Yöneticisi : Doç. Dr. Esma Gaygısız
Ortak Tez Yöneticisi : Doç. Dr. Hamdullah Yücel

Aralık 2023, 116 sayfa

Günümüzde ağların yapıları, oluşumları ve kararlılıklarının analizi, önem kazanmakta ve ilgi çekmektedir. Bu tezde, özel bir ağ oluşum oyunu ortaya konmakta ve yeni bir kararlılık konsepti önerilmektedir. Önerilen oyun, ağ bağlantılarıyla giderlerini azaltarak getirilerini arttıran firmalar arasında oynanmaktadır. Çoklu bağlantı yaklaşımında, bireylerin bağlantılarını değiştirerek en yüksek getiriye ulaşması bir karışık tamsayılı optimizasyon problemi çözülerek bulunur. Çok sayıda bağlantının optimal kombinasyonunu bulmak karmaşık bir problem yaratır. Bu problemi çözmek için probleme özel bir interpolasyon yöntemi ile dallandırma ve sınırlandırma yönteminin bir modifikasyonu sunulmaktadır. Önerilen modelin ağ oluşumunu tasvir etmekteki yeterliliği, farklı parametreler kullanılarak yapılan simülasyonlarla araştırılmaktadır. Bu simülasyonlar tezde önerilen modelde çoklu bağlantı kararlılığının, ikili kararlılık olmadığında bile bulunabileceğine işaret etmektedir. Bu sonuca dayanarak, yeni bir kararlılık konsepti olan ağlar için çoklu bağlantı kararlılığını öne sürüyoruz.

Anahtar Kelimeler: Ağ Oluşumu, Ağ Dengesi, Karışık Tamsayılı Doğrusal Olmayan Optimizasyon

This thesis is dedicated to a happier tomorrow.

ACKNOWLEDGMENTS

I would like to express my gratitude to my supervisor Assoc. Prof. Dr. Esma GAYGISIZ, who materialized my enthusiasm and interest into this thesis, and my co-supervisor Assoc. Prof. Dr. Hamdullah YÜCEL, whose patience and understanding helped me a lot while putting this thesis together. I would also like to thank members of my thesis defence committee for their insightful comments and discussions.

Moreover, I would like to thank my friends. After studying networks for about two years, I can confidently say that their support made this process easier. In particular, I want to thank my girlfriend İzel ÇELİK, to whom the burden of supporting me mostly fell over the past two years.

Finally, I would like to acknowledge my family. I appreciate all of their support. However, I feel especially appreciative of the support of my parents, who made my graduate studies possible, and my grandparents, whose ceaseless prayers enabled me to finish this thesis.

TABLE OF CONTENTS

APPENDICES

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

$\mathbb{R}^+$	Positive real Numbers.
J	The number of nodes or firms in a system.
$\mathbf{I}$	Identity matrix.
N_i	The number of firms that are directly or indirectly connected to firm i.
D_i	The number of direct connections to firm i.
q	Quantity of produced goods.
P	Price for a unit of good.
β_i	Linear cost coefficient of production for the good for firm i.
γ_i	Quadratic cost coefficient of production for the good for firm i.
$c_{i,j}$	The cost coefficient of sustaining a connection from firm i to firm j.
d_i	The cost coefficient for the number of firms directly or indirectly connected to firm i.
$\pi_i()$	Profit function for firm i.
G	The adjacency matrix of the network of firm interactions.
sgn()	The sign function.
$\delta_{i,k}$	The choice variable for firm i to connect with k.
$\mathcal{G}_i(\delta_i, G)$	The function that alters G based on choices given in δ_i vector.
$\mathcal{R}_i()$	The function that yields the difference in profit of firm i due to choices.
$B_k(\delta_i, G)$	The function used for checking the component condition.
n	The continuous equivalent of N_i.
m	The continuous equivalent of D_i.
PW	Pairwise.
MC	Multiple connectivity.
MINLP	Mixed integer non-linear programming.
MBB	Modified branch and bound.

CHAPTER 1

INTRODUCTION

This thesis is a study on network formation games. We focus on short-term decision-makers whose profits depend on their network structure and propose a new approach to modelling their choices. This new approach allows us to compare how players' profit varies when they can selectively alter all their direct connections. As a result, we are able to capture network formation choices to maximise profit more accurately. While this thesis focuses on a static game with short-term thinking players, we consider our approach to be applicable to the more general study of network formation games.

The observation that individuals with similar interests interact with others is ubiquitous. From children in the playground to the multi-national alliances, we can see that when left to their own devices, individuals tend to form groups by selectively including and excluding others. As such, the attempt to observe, quantify, and model such interactions has a rich history. One of the earlier notable attempts was made by L. Katz, where the popularity of high school students was examined not just by the sheer quantity of votes but by who picks who individually [34]. Another earlier work was carried out by J. A. Barnes, who studied social organisation in the small Norwegian Island Parish of Bremnes [5]. In later years, we see that the study of interactions of individuals expands to many other fields [4, 8, 15, 40].

The nature of relationships between individuals is often captured using graph theory when studying networks. One of the earliest examples of this methodology can be seen in the work of Harary and Norman [25]. Using the graph theory, various network properties, such as centrality [21] and power [10], can be measured. Moreover, using the graph theory and network measures, whether an individual is better off by participating in or abstaining from a particular network structure can be quantified. This approach leads to the studies on endogenous network formation, that is, individuals forming a network with their own choice [1, 42].

In this study, the formation of networks is considered from a game theoretical angle, where each player's payoffs depend on other players in their network and how they are connected. Broadly, payoffs may increase or decrease based on the choices of others in a player's network, which are called games of strategic complements and substitutes, respectively. Then, depending on the nature of the game, the player may adjust the network structure to increase their payoff, and if no higher payoff is possible for any moves for any players, the network is said to be stable [33]. The profit for players may be the result of the intrinsic value of their connections [9, 12, 26, 31], through choices and payoffs of others in their network [3, 11, 32], or through their ability to collectively respond to external events such as risk management [2]. Our model uses direct and indirect connections while calculating the payoff. In particular, we relate these network properties to the production costs. Studies relating network effects to production cost reduction via technology transfer and know-how can be found in [17, 23, 35, 36]. Moreover, we consider an additional cost induced by being connected to a network in terms of direct connection and the size of total connections similar to the study [24].

In the literature, there are different approaches to how a network formation game may play out and how players may increase their profit. One approach is presented by Myerson [37], where each player gives a list of alterations they wish, and if both parties wish to connect, a connection is made. This approach treats network formation similar to a matchmaking problem where a particular network may only be stable if no pair of players prefers altering their connections. A study that considers college admissions in a similar light is [22]. More recently, Jackson and Wolinsky present pairwise approach to network formation, where each connection is considered as separate moves by its parties [28]. Later studies developed this notion to include long-term thinking players that make multiple moves or those that anticipate responses from other players to increase their profit, such as [19, 29, 30]. These studies consider network formation not as a process that starts from one network and reaches another but as an analysis of whether an existing structure is stable or not. And we see that stability is understood as players' unwillingness to deviate from the structure they are in. Studies exist that do consider formation as a dynamic game, such as [43, 44]. However, in this thesis, we consider a static network formation game and focus on conditions required for the stability.

As discussed in [27], pairwise formation may yield network structures where players are stuck in a network that does not yield the greatest possible profit for some players but is still stable, as no player can improve their profit by a single pairwise move. In such systems, it may be possible for a player to reach a more profitable configuration over multiple alterations to its connections. We consider this to reflect network formation insufficiently, and therefore, in this study, we consider a new kind of network formation game where individuals have the choice of altering all their connections as a single move in order to maximise their profits.

In order to find the optimal moves for players where every possible connection is evaluated at once, we propose a method to express individual connections in terms of binary (yes or no) choice variables and write the profit maximisation problem as a mixed integer optimisation problem. Incorporating mixed-integer optimisation with the game theory in order to find the best choices is not a new concept. Studies exist that suggest such methodology for smart power grid [47], waste management [13], or supply chain management [46]. Moreover, studies such as [38, 41] use mixed integer programming to find more efficient networks. However, using mixed integer programming to find optimal choices in a network formation game is a new concept that we aim to contribute to the game theory literature with this study.

Including introduction and conclusion, this thesis is organised into five chapters. In Chapter 2, we compare pairwise and multiple connectivity approaches to network formation games. Then, we propose the Component Profit Model to relate being a part of a network to profit and set up a game that describes how players may make choices that alter the network to maximise their profit. Then, in Chapter 3, we discuss how our model can be solved numerically, along with our method of interpolating discrete choice variables and our modifications to existing methods to solve mixed integer optimisation problems. This discussion is followed by Chapter 4 where we present an example to further illustrate the merits of using multiple connectivity approach to pairwise approach and simulate our model with different parameters to observe how the networks are formed by playing the game we set up. Finally, in Chapter 5, we present our conclusions for this study and remark on how future work in this field may be pursued.

CHAPTER 2

NETWORK FORMATION WITH MULTIPLE CONNECTIVITY

2.1 Introduction

This thesis establishes a specific network formation game and suggests a network stability concept with the multiple connectivity approach. The game is among firms forming networks to reduce their costs. The firms' discrete connection choices in the formation of stable networks are explored in this chapter.

In Section 2.2, we explain some basic network concepts that we use. Then, in Section 2.3 we discuss pairwise stability as introduced by Jackson and Wolinsky [31] and present our alternative, the multiple connectivity approach. Next, in Section 2.4, we introduce firms with their incentives to connect to a network or to disconnect from a network: the Component Profit Model. Later in Section 2.5, we unite the Component Profit Model and multiple connectivity approach to discuss the stability characteristics of networks. This is followed by Section 2.6, where we consider a benchmark example consisting of three firms that operate with the Component Profit Model and examine various network configurations for stability. Finally, we summarise our model in Section 2.7.

2.2 Basics of the Graph Theory

Graph theory is commonly used to study networks, and it provides methods to analyse how different individuals are related to each other. The individual is a general term that may describe anything from parts of a city to atoms in a molecule, and the relations may mean any interaction individuals have with each other. We take each profit-maximising firm as a decision-making individual in this study. We use the concepts from graph theory to describe an individual's direct connections in a network and the total number of individuals in that network influencing the individual's payoff.

2.2.1 Network Properties

The building blocks of the network formation with a graph theoretical approach are discussed in this section by using the books Introduction to Graph Theory [45] and Social and Economic Networks [27]. The former is for the graph theoretical concepts, and the latter is for their applications in network games. We begin with the definitions for node and edge:

- Node: A single point in a network that connects to others (vertex).
- Edge: Connections between nodes (line).

How the nodes are connected, in particular, can be described using a set of edges or an adjacency matrix. We opt for the latter as it makes computation more straightforward. From here on, we define the adjacency matrices of networks with J nodes as $G \in \{0, 1\}^{J \times J}$. G is comprised of individual connections between firms, which are denoted as $G_{i,j} = 1$ if a connection exists between i and j and 0 if it does not.

Connections in a network can be one-sided or two-sided (unidirectional or bidirectional). If it is the latter, the matrix G is symmetric. For this study, we focus on the two-sided connections, and we assume $G_{i,i} = 0$, thus implicitly assuming there is no such thing as a node as being connected to itself. Going further, we borrow more concepts from the graph theory:

- Walk: A sequence of edges where each edge is connected to the next by one of its nodes and the previous one with the other.

- Path: A walk where an edge is not repeated.

- Cycle: A walk that starts and ends on the same node.

Moreover, we can say that a graph is **connected** if a path exists between every node in a graph and disconnected otherwise. Disconnected graphs are obtained by joining smaller connected graphs. These connected graphs are called **components**. A component may be a single isolated node or the entire graph (if all nodes are connected). If a component contains no cyclic paths, it is called a tree, and if all components of a graph are trees, it is called a **forest**.

Additionally, we use the following definitions:

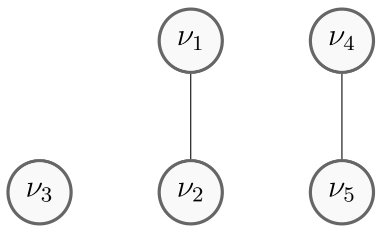
- System: The collection of all nodes and edges that we are considering.

- Component Size: The number of nodes in a component.

- Direct Connections: All the nodes a firm can reach by a single edge walk (degree of a node).

Next, we can define two vectors, $D \in \{0, 1, \dots J-1\}^J$ that denotes the number of direct connections, and $N \in \{1, 2, \dots J\}^J$ which is the component size. If a node is not connected to any other, its number of direct connections is zero, and we take its component size as one.

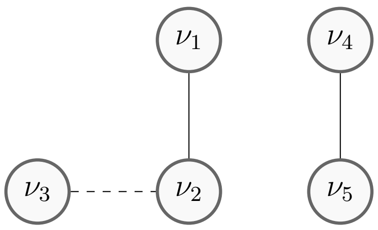
To better explain these concepts, we can look at an example with five nodes denoted as ν_i, where $i \in 1 \dots 5$ refers to node i in Figure 2.1.



$$\equiv G = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}, D = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}, N = \begin{bmatrix} 2 \\ 2 \\ 1 \\ 2 \\ 2 \end{bmatrix}$$

Figure 2.1: A system with five nodes and two edges.

In the system given in Figure 2.1, there are three components. Nodes 1 and 2 are the first component, node 3 is a component of one node, and the rest are the elements of the third component.

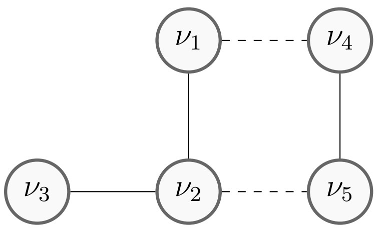


$$\equiv G = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}, D = \begin{bmatrix} 1 \\ \mathbf{2} \\ \mathbf{1} \\ 1 \\ 1 \end{bmatrix}, N = \begin{bmatrix} \mathbf{3} \\ \mathbf{3} \\ \mathbf{3} \\ 2 \\ 2 \end{bmatrix}$$

Figure 2.2: A system with five nodes, with the new connection shown .

Now, let us assume node 3 is connected to node 2. As seen from Figure 2.2, this connection alters the direct connections of involved parties (2,3) but also increases total connections for node 1. This connection also merges two of the components, thus reducing the number of distinct components from three to two.

Now, we convert this system into a single component by connecting 1 to 4 and 2 to 5.

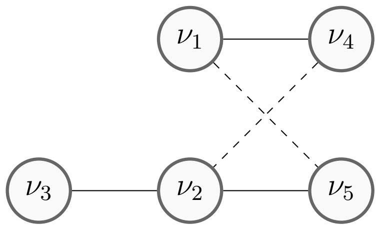


$$\equiv G = \begin{bmatrix} 0 & 1 & 0 & \mathbf{1} & 0 \\ 1 & 0 & 1 & 0 & \mathbf{1} \\ 0 & 1 & 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 & 0 & 1 \\ 0 & \mathbf{1} & 0 & 1 & 0 \end{bmatrix}, D = \begin{bmatrix} \mathbf{2} \\ \mathbf{3} \\ 1 \\ \mathbf{2} \\ \mathbf{2} \end{bmatrix}, N = \begin{bmatrix} \mathbf{5} \\ \mathbf{5} \\ \mathbf{5} \\ \mathbf{5} \\ \mathbf{5} \end{bmatrix}$$

Figure 2.3: A system with five nodes, with the new connection shown with dashed lines.

In Figure 2.3, we see that N is equal to 5 for all nodes, but the number of direct connections varies between nodes.

While G defines a system uniquely on its own, N or D are not unique for G. To demonstrate this, we can shuffle connections without adding and removing any.



$$\equiv G = \begin{bmatrix} 0 & \mathbf{0} & 0 & 1 & \mathbf{1} \\ \mathbf{0} & 0 & 1 & \mathbf{1} & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & \mathbf{1} & 0 & 0 & \mathbf{0} \\ \mathbf{1} & 1 & 0 & \mathbf{0} & 0 \end{bmatrix}, D = \begin{bmatrix} 2 \\ 3 \\ 1 \\ 2 \\ 2 \end{bmatrix}, N = \begin{bmatrix} 5 \\ 5 \\ 5 \\ 5 \\ 5 \end{bmatrix}$$

Figure 2.4: A system with five nodes, with the new connection shown with dashed lines.

We see in Figure 2.4 that even though the connection matrix, G, has changed, vectors D and N have not.

2.2.2 Connection Numbers

In the previous section we show how G affects vectors D and N. In this section, we aim to quantify this effect. For D, we can simply write:

$$D_i = \sum_j G_{i,j}. \tag{2.1}$$

This expression is consistent with the fact that a row of an adjacency matrix contains all the direct connections of a corresponding node.

When expressing N_i, however, we need to devise a more elaborate scheme. To start with, we note that G, as an adjacency matrix, contains all the first-degree (direct) connections of a node. Another way of interpreting it is if $G_{i,j} = 1$, node i can reach node j with a single step. Furthermore, $(G^n)_{i,j} = k$ means node i can reach node j by n steps by k different walks.

With this in mind, let us define $H = G + \mathbf{I}$, where $\mathbf{I}$ is the identity matrix. Unlike the adjacency matrix, where no node is connected to itself, this structure defines a system where all the nodes are self-connected. From this notion, we state the following theorem.

Theorem 1. *If $G \in \{0,1\}^{J \times J}$ is an adjacency matrix of a bidirectional graph and $H = G + \mathbf{I}$, where $\mathbf{I}$ is the identity matrix of relevant size and if $H_{i,j}^{J-1} > 0$, two nodes* *are in the same component.*

Proof. We know that n^{th} degree of an adjacency matrix $H^n_{i,j}$ counts possible walks from i to j. However, as all nodes are self-connected, H^n also includes all walks from H^{n-1} and one step to the node itself. By induction, this means for H^n, there is at least one walk from every earlier power. Thus, if $H^n_{i,j} > 1$, we can say that there is at least one walk that starts with i to i and then contains the shortest walk from i to j.

Then, in a system with J nodes, two nodes cannot be further than $J - 1$ steps apart. It follows that $H^{J-1} = 0$ only if two nodes are not connected by any walk, as such path would be included in the sum of paths for $H^{J-1}_{i,j}$. Thus it is sufficient to look for $(J-1)^{th}$ power of H to infer a connection and check if the relevant element is greater than zero. $\square$

We can then claim that if we count $H^{J-1}_{i,j} > 0$ for the values of i, we find total nodes in a component. It follows that we can compute N_i by:

$$N_i = \sum_j \text{sgn}((G + \mathbf{I})^{J-1})_{i,j}, \tag{2.2}$$

where G is the adjacency matrix of the system, $\mathbf{I}$ is the identity matrix, sgn() is the sign function applied element-wise to its input matrix, and J is the number of nodes.

As an example, we reconsider the network in Figure 2.2, where the network is given by:

$$G = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}. \tag{2.3}$$

We can rearrange the formula we derived in (2.2) to find the component size vector:

$$N = \text{sgn}((G + \mathbf{I})^{J-1})\mathbf{1}, \tag{2.4}$$

where $\mathbf{1}$ is a vector of ones. When we apply the formula we derived to $G + I$, we find:

$$N = \text{sgn}\left(\begin{bmatrix}1&1&0&0&0\\1&1&1&0&0\\0&1&1&0&0\\0&0&0&1&1\\0&0&0&1&1\end{bmatrix}^4\right)\begin{bmatrix}1\\1\\1\\1\\1\end{bmatrix} = \text{sgn}\left(\begin{bmatrix}9&12&8&0&0\\12&17&12&0&0\\8&12&9&0&0\\0&0&0&8&8\\0&0&0&8&8\end{bmatrix}\right)\begin{bmatrix}1\\1\\1\\1\\1\end{bmatrix}$$

$$= \begin{bmatrix}1&1&1&0&0\\1&1&1&0&0\\1&1&1&0&0\\0&0&0&1&1\\0&0&0&1&1\end{bmatrix}\begin{bmatrix}1\\1\\1\\1\\1\end{bmatrix} = \begin{bmatrix}3\\3\\3\\2\\2\end{bmatrix}. \tag{2.5}$$

This result is equal to the component size vector we found in Figure 2.2 by inspecting the network.

2.2.3 Miscellaneous Network Definitions

Before ending this section, we provide some miscellaneous definitions we use while talking about networks in this study. First, while talking about nodes, we use the definitions:

- Isolated Node: A node that has no connections.
- Leaf Node: A node that has only one connection to its component.
- Inner Node: A node in a component that is not a leaf.

In this section, we mainly encounter forest graphs. A forest may include components that are pure chains, pure stars, or a mix, which can be defined as:

- Chain Component: A component that is a tree where no node has more than two connections (as given in Figure 2.5).
- Star Component: A component with a single central node and other nodes connected to it (as given in Figure 2.6).

- Mixed Component: A component that is a tree but is not a pure star or a pure chain (as given in Figure 2.7).

While these definitions can be applied to components of all sizes, they are more meaningful for components with more than three nodes. For the networks of smaller components, we also use the following expressions:

- Unconnected Network: A network where all of the nodes are isolated.
- Network of Pairs: A network where no node has more than one connection.

Finally, the node labelling of a graph is how each node is labelled, and two graphs are **isomorphic** if a one-to-one mapping of node labels can be defined between the two while maintaining the number of connections to each corresponding node. We note that while the values of N and D depend on the network structure, the exact index of those values depend on the labelling, and therefore isomorphs of a network has the same N and D values in different order.

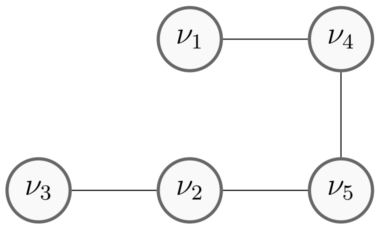


Figure 2.5: A network with a single chain component.

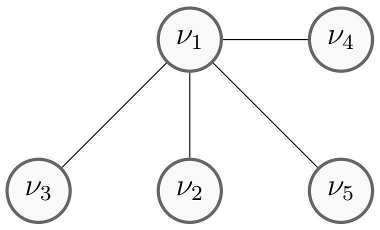


Figure 2.6: A network with a single star component where ν_1 is the central node.

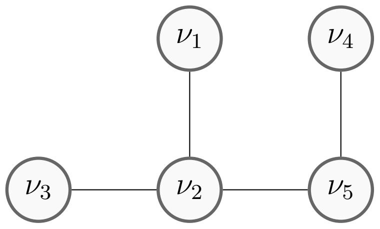


Figure 2.7: A network with a single mixed component.

2.3 Multiple Connectivity Approach

In this section, we introduce the multiple connectivity approach, which is our primary contribution to the literature on network formation. The principle novelty of this approach is offering firms a choice to connect with all the other firms available and asses profit over all the possible connections the firm can make. We first show the effects of altering the network for players and the concept of stability with regard to network formation games. Then, we present our approach and how it differs from the literature.

We discuss a situation with J payoff-maximising players whose payoff depends on the network G that they are in. The model we use to find this payoff is discussed in Section 2.4. For this section, however, we consider an arbitrary payoff function of $u_i(G)$ for firm i. We can say that firm i prefers G to G' if:

$$u_i(G) > u_i(G'). \tag{2.6}$$

This preference implies that if firm i finds itself in G' and may alter the network to reach G, it would. We can expand this simple notion to the concept of stability. Dutta and Mutuswami [19] define weakly and strongly stable networks as networks in which players cannot increase their payoff by altering as individuals or through co-ordination, respectively. For this study, we focus on network structures where firms' choices only extend to their direct connections. Therefore, we focus on the weak stability.

2.3.1 Pairwise Approach

The notion of weak stability in network formation studies is often associated with pairwise stability based on the work of Jackson and Wolinsky [31]. The approach followed in that study is to consider each player pair separately and assess whether its parties increase their payoffs or not via altering their direct connection. We refer to this approach as a pairwise (PW) approach.

We can show a pairwise alteration to the connection between i and k for G as $G^{(i,k)} \equiv G^{(k,i)}$. If no pairwise alteration to G is preferable to i, it will not choose to deviate from it. This can be shown as:

$$u_i(G^{(i,k)}) \leq u_i(G) \quad \forall i \in 1, \ldots, J, \;\; k1, \ldots, J-1, \tag{2.7}$$

no player prefers to pairwise deviate from G.

2.3.2 Consent and Pairwise Stability

The discussions on the pairwise approach are complemented with a discussion on consent as exemplified by [31]. While weak stability, by definition, means players do not coordinate, it is assumed that network connections are two-sided, and both sides must at least not disagree in order for a connection to be possible. To better illustrate this, we can consider a network between only two players. There are two possible networks with two players, connected and not connected. We can call these cases G and G' respectively, and represent them as:

$$G = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \; G' = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}. \tag{2.8}$$

We can further say that $G^{(1,2)} = G'$, and $G'^{(1,2)} = G$, that is to say, G and G' are pairwise deviations of each other. Let us consider payoffs for two players as u_1 and u_2 given in Table 2.1. It is clear that both players obtain higher payoffs when their connections are G, hence they prefer G to G'. Then, for these payoffs, we can say that no player prefers to deviate pairwise from G.

For an alternate result, however, we can consider the payoffs u_1' and u_2' given in Table

2.2. Here, player 1 prefers G', and player 2 prefers G. When players start from either of the cases, at least one would prefer making a pairwise alteration.

Table 2.1: Payoffs of two players with networks G and G'.

	u_1	u_2
G	1	1
G'	0	0

Table 2.2: Alternative payoffs of two players with networks G and G'.

	u'_1	u'_2
G	0	1
G'	1	0

For payoff-maximising players, it can be further asserted that they would only consent to form a connection if their payoff does not decrease due to it. We call this the consent condition. We can further assert that players can break connections unilaterally, and therefore, no consent condition is required for them to do so.

With the consent condition imposed, we can reconsider the case given in Table 2.2. Clearly, player 1 would still deviate unilaterally from G. However, player 2 needs the consent of player 1 to deviate from G'. Since player 1 would not consent to make a connection that leaves it worse off, we can say that no player would pairwise deviate from G' when the consent condition is enforced.

The preference of not pairwise deviating from a network G with the consent condition can be generalised as:

$$\begin{aligned}&\big(u_i(G) > u_i(G^{(i,k)})\big) \vee \big(u_k(G) > u_k(G^{(i,k)})\big)\\ \vee&\Big(\big(u_i(G) = u_i(G^{(i,k)})\big) \wedge \big(u_k(G) = u_k(G^{(i,k)})\big)\Big) \qquad \forall\, G_{i,k} = 0. \qquad (2.9a)\\ &\big(u_i(G) \geq u_i(G^{(i,k)})\big) \wedge \big(u_k(G) \geq u_k(G^{(i,k)})\big) \qquad \forall\, G_{i,k} = 1. \qquad (2.9b)\end{aligned}$$

A system G that meets these conditions can also be referred to as pairwise stable [31].

2.3.3 Problems with Pairwise Stability

Pairwise stability is a useful tool in analysing networks. However, considering connections one at a time creates some problems. We can illustrate one such problem by considering systems:

$$G = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, G' = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \tag{2.10}$$

and G'' is any bidirectional adjacency matrix in $\{0,1\}^{3\times 3}$ where $G'' \neq G$ and $G'' \neq G'$ and their respective payoffs are given in Table 2.3. It requires three pairwise alterations for G to reach G', one to break the 1,2 connection and the other to make 1,3 connection, which is $G^{(1,2)}$ and $G^{(1,3)}$ respectively. Similarly, reaching G from G' requires two alterations. Either of the alterations, on its own, cannot increase any player's payoff. Therefore, both G and G' are pairwise stable. However, realistically, we expect player 1 to make a long-term calculation and make two pairwise alterations to the system G. This preference cannot be captured with the notion of pairwise stability.

Table 2.3: Payoffs of three players.

	u_1	u_2	u_3
G	1	1	0
G'	2	0	1
G''	0	0	0

Another problem with this configuration can be found when we consider a fully connected system. Altering a single connection can only create chains, which are also not beneficial for any player with the given payoff values. This means (2.9) holds for the fully connected system, and it is pairwise stable. In such a situation, player 1 may sever the connection with player 2 and hope player 3 will do the same. However, this requires players to coordinate and make long-term thinking decisions. A stability notion that encompasses such decisions is presented by Jackson and Nouweland [28]. In our study, we posit that it is also possible to tackle these problems within the scope of non-coordinating decision-makers. To that end, we introduce a way for players to consider adding or deleting connections simultaneously.

2.3.4 Multiple Connectivity

First, we need to express all the networks player i may generate by altering their direct connections from a prior network G. To do this, we define the choice variable $\delta_{i,k} \in \{0,1\}$ to describe the connection from i to k (we assume $k \neq i$ since i cannot connect to itself). Using this variable and the prior system G, we can define the function:

$$\mathcal{G}_i(\delta_i, G)_{a,b} = \begin{cases} \delta_{i,k}, & \text{if } (a = i \wedge b = k) \vee (b = i \wedge a = k), \\ G_{a,b}, & \text{otherwise.} \end{cases} \tag{2.11}$$

Then, all possible alteration i can make to its direct connections is expressed by different values δ_i vector takes. With this, we can write the best possible network obtainable by i by altering G as:

$$u_i(\mathcal{G}_i(\hat{\delta}_i, G)) \geq u_i(\mathcal{G}_i(\delta_i, G)) \quad \forall\, \delta_i, \tag{2.12}$$

where $\hat{\delta}_i$ is the best alteration of direct connections player i can make. To find such a δ_i, we solve a mixed integer optimisation problem as follows:

$$\begin{aligned} &\max_{\delta_i} u_i(\mathcal{G}_i(\delta_i, G)), \\ &s.t.\ \delta_{i,k} \in \{0,1\} \quad \forall\, k. \end{aligned} \tag{2.13}$$

The possible values δ_i can take is a finite set. Therefore, this problem is guaranteed to have a solution. However, this solution does not have to be unique. Nevertheless, similar to the pairwise deviation, we can say that a player would attempt to alter its connections only if the new network they propose (with $\hat{\delta}_i$) improves its payoff. This means that either player is already in an optimal state of connections and would not benefit from altering its network to reach the same payoff or is not in an optimal state and has at least one alternative it would prefer.

2.3.5 Consent for Multiple Connectivity

We must also address the issue of consent. With pairwise connections, we say that if a connection is added, one side must benefit, and the other must at least not be worse off with the new connection as given in (2.9a). And when breaking connections, at least one side must benefit, as given in (2.9b). Similarly, when a player proposes an alteration, it needs to ensure the players they are proposing to connect do not lose payoff from the new network. We do this by comparing the payoffs of the newly proposed network to the prior network for every directly connected player. This idea can be written as:

$$\mathcal{R}_k(\delta_i, G) = u_k\big(\mathcal{G}_i(\delta_i, G)\big) - u_k\big(G\big), \tag{2.14}$$

and the multiple connectivity consent condition for player i to connect with k as:

$$\mathcal{R}_k(\delta_i, G)\delta_{i,k} \geq 0. \tag{2.15}$$

The difference in the payoffs for players is multiplied by the choice of direct connection to make sure every directly connected player consents, while the payoff difference for the other players is ignored. Having established the role of consent in the multiple connectivity, we can rewrite the optimisation problem as follows:

$$\begin{aligned} &\max_{\delta_i} u_i(\mathcal{G}_i(\delta_i, G)), \\ &\quad s.t.\ \mathcal{R}_k(\delta_i, G)\delta_{i,k} \geq 0 \quad \forall\, k, \\ &\qquad \delta_{i,k} \in \{0, 1\} \quad \forall\, k. \end{aligned} \tag{2.16}$$

Finally, when the solution to this problem fails to produce a greater payoff for a player, we can say that the player would not propose any alterations. And when no player can propose a better system, we can call that system multiple connectivity stable (MC stable).

2.3.6 Comparing PW and MC Stability

Our aim in developing a multiple connectivity approach is to address problems in a pairwise approach while retaining the short-sightedness and non-coordination of players. Multiple Connectivity Stability is still a form of weak stability based on the definitions given by Dutta and Mutaswani [19]. However, within the constraints of non-coordination and short-sightedness, it gives players much greater freedom to increase their payoff. Indeed, for J players, a player has $J - 1$ PW alterations and 2^{J-1} MC alterations. The additional options make it less likely for a player to get stuck in less beneficial situations that are still PW stable. To demonstrate this, let us reconsider the case given in Table 2.3. We start with G and write the networks attainable by player 1 as:

$$\mathcal{G}_1(\delta_1, G) = \begin{bmatrix} 0 & \delta_{1,2} & \delta_{1,3} \\ \delta_{1,2} & 0 & 0 \\ \delta_{1,3} & 0 & 0 \end{bmatrix}. \tag{2.17}$$

The payoffs associated with the connection values are given in Table 2.4. It is clear that the highest payoff attainable by player 1 is 2, and player 3 would consent to this connection as they stand to increase their payoff by 1 from this alteration. Thus, we can say that G is not MC stable.

Table 2.4: Payoffs of three players based on δ_1.

$\delta_{1,2}$	$\delta_{1,3}$	u_1	u_2	u_3
0	0	0	0	0
1	0	1	1	0
0	1	2	0	1
1	1	0	0	0

Next, we consider G' and start with the payoffs attainable by player 1. Once again, we start by writing possible networks Player 1 can generate.

$$\mathcal{G}_1(\delta_1, G') = \begin{bmatrix} 0 & \delta_{1,2} & \delta_{1,3} \\ \delta_{1,2} & 0 & 0 \\ \delta_{1,3} & 0 & 0 \end{bmatrix}. \tag{2.18}$$

This expression is identical to (2.17). It follows that 2.4 also shows the payoffs for this situation. However, we already know that player 1 benefits most when they make a single connection to player 3, which is the network G'. It follows that player 1 does not benefit by altering the network. We proceed by writing the case for player 2:

$$\mathcal{G}_2(\delta_2, G) = \begin{bmatrix} 0 & \delta_{2,1} & 1 \\ \delta_{2,1} & 0 & \delta_{2,3} \\ 1 & \delta_{2,3} & 0 \end{bmatrix}. \tag{2.19}$$

Table 2.5 shows the payoffs corresponding to these decisions. These values show that Player 2 cannot propose any network to improve its payoff.

Table 2.5: Payoffs of three players based on δ_2.

$\delta_{2,1}$	$\delta_{2,3}$	u_1	u_2	u_3
0	0	2	0	1
1	0	0	0	0
0	1	0	0	0
1	1	0	0	0

Finally, we can look at player 3, whose decisions can be represented as:

$$\mathcal{G}_2(\delta_2, G) = \begin{bmatrix} 0 & 0 & \delta_{3,1} \\ 0 & 0 & \delta_{3,2} \\ \delta_{3,1} & \delta_{3,2} & 0 \end{bmatrix}. \tag{2.20}$$

The corresponding payoffs are given in Table 2.6, where it is also evident that player 3 cannot propose any alterations to improve their payoff. It follows that G' is MC stable.

Table 2.6: Payoffs of three players based on δ_3.

$\delta_{3,1}$	$\delta_{3,2}$	u_1	u_2	u_3
0	0	0	0	0
1	0	2	0	1
0	1	0	0	0
1	1	0	0	0

2.4 Component Profit Model

In our study of network formation, we take profit-maximising firms as the decision-makers that form the network. In this section, we consider an individual firm and discuss how it alters its interactions with other firms to form a profitable network.

Firms produce goods at certain costs and sell them to earn the difference as profit [39]. In this study, firms make their production decisions in order to maximise profits and interact with other firms in a certain market in terms of the quantities they supply and the network they form.

In the proceeding analysis, we first discuss the production decisions of the firms.

2.4.1 Production Decisions of the Firms

We assume all the firms in our study produce a single good and compete in a market where they cannot affect the price by changing the amount they produce. The price may be controlled by an external force such as a government, or the firms in question may represent a negligible quantity of total production in their market so that each firm takes its effect on the market price as negligible. This means their revenue linearly depends on the market price $P \in \mathbb{R}^+$.

Such firms may still compete by lowering their unit production costs to maximise their profit. We can attribute the total costs and revenues firms face to two sources: internal operations and external operations. Internal operations are related to producing and selling the good, while external costs are related to being connected to other firms.

We posit that by forming connections with other firms that produce the same good, a firm can decrease the cost it faces per production. This may happen by sharing resources or know-how. In order to capture this benefit, we divide the cost parameter $\beta_i > 0$ by the number of firms in the firm $i's$ component (which we denoted by N_i in Section 2.2) to find $\frac{\beta_i}{N_i} q_i$. This represents the advantage of being connected to other firms. Additionally, we formulate $\gamma_i q_i^2$ as a cost that only depends on the quantity produced by the firm. We define $\mathcal{C}_i(q_i, N_i)$ as the cost of producing q_i and write as:

$$\mathcal{C}_i(q_i, N_i) = \frac{\beta_i}{N_i} q_i + \gamma_i q_i^2. \tag{2.21}$$

With this expression, we aim to capture how being connected reduces the production cost so that an unconnected firm's profit changes more dramatically by forming a connection. In contrast, a firm in a component of 100 firms is barely affected by the 101st connection. When the firm is not connected to others, i.e., $N_i = 1$, and the expression becomes the standard production cost function:

$$\mathcal{C}_i(q_i, N_i) = \beta_i q_i + \gamma_i q_i^2. \tag{2.22}$$

Moreover, as the production amount increases to high levels, the quadratic cost ensures that firms with similar properties in different components face similar costs, therefore keeping the optimal production and profit values comparable.

We can merge Pq_i and (2.21) and find the total profit from the internal operations:

$$Pq_i - \mathcal{C}_i(q_i, N_i) = Pq_i - \frac{\beta_i}{N_i} q_i - \gamma_i q_i^2. \tag{2.23}$$

Firm i's profit maximisation problem is solved in two steps. First, as step 1 given that Firm i is functioning in a market with N_i firms, Firm i's profit maximising quantity of production is found. Then, as step 2, Firm i's network connections are analysed based on the found quantity.

In the first step, we find the profit-maximising quantity of production. We define the profit generated by q_i for a given $\overline{N}_i$ as $\mathbf{P}_i(q_i)$ and write the first and second derivatives:

$$\frac{d\mathbf{P}_i(q_i)}{dq_i} = P - \frac{\beta_i}{\overline{N}_i} - 2\gamma_i q_i, \tag{2.24a}$$

$$\frac{d^2\mathbf{P}_i(q_i)}{dq_i^2} = -2\gamma_i. \tag{2.24b}$$

Since $\gamma_i > 0$ we can say that:

$$\frac{d^2\mathbf{P}_i(q_i)}{dq_i^2} = -2\gamma_i < 0 \tag{2.25}$$

Hence, the production function is strictly concave in $q_i \in \mathbb{R}^+$. Hence, the first-order condition for profit maximisation yields:

$$0 = P - \frac{\beta_i}{\overline{N}_i} - 2\gamma_i \hat{q}_i, \tag{2.26}$$

$$\hat{q}_i = \frac{\overline{N}_i P - \beta_i}{2\overline{N}_i \gamma_i}. \tag{2.27}$$

While mathematically, this expression may yield negative results, which do not make sense as an amount of production, we implicitly assume firms can profit while not connected ($N_i = 1$) by assuming $P > \beta_i$. With this, $N_i \geq 1$, $\hat{q}_i$ is always positive.

Using (2.27) we find the optimal profit level as:

$$\mathcal{P}_i(N_i) = \frac{\overline{N}_i P - \beta_i}{2\overline{N}_i \gamma_i}. \tag{2.28}$$

By maximising this expression, we can find the optimal component size for internal profit. However, a firm in a network may also face some costs associated with its network properties. Thus, to find the total profit function, we also need to discuss the cost of external operations.

2.4.2 The Cost from Forming and Maintaining Connections

The external operations of a firm are related to establishing and sustaining connections with other firms. Now, we consider how the costs associated with being connected to a network are described.

We assume that Firm i experiences a connection cost with each Firm $j \neq i$ that it is directly connected with as $c_{i,j} > 0$. We can multiply these values with the adjacency matrix and write the total cost of direct connections as:

$$\sum_j c_{i,j} G_{i,j}. \tag{2.29}$$

Furthermore, being in a larger component should incur an extra cost even with the same number of connections for a firm. To capture this, we define another coefficient d_i to describe the cost firm i faces with $N_i - 1$ connections, which is the number of firms in a component minus one, as $N_i = 1$ means a firm is not connected to others.

From these notions, we can derive the following expression:

$$\mathcal{N}_i(N_i, G) = d_i(N_i - 1) + \sum_j G_{i,j} c_{i,j}. \tag{2.30}$$

Here, the coefficients d_i and $c_{i,j}$ can be modified to match the exact conditions in the modelled system.

Assuming the connection costs only vary for firms individually and are the same regardless of the other party, that is, $c_{i,j} = c_i$, we can further simplify this expression by:

$$\mathcal{N}_i(N_i, G) = d_i(N_i - 1) + c_i \sum_j G_{i,j}. \tag{2.31}$$

We can further denote the direct connections of a firm as $D_i = \sum_j G_{i,j}$ (as given in Section 2.2) and substitute that to find the expression for the cost of external operations of a firm in our model:

$$\mathcal{N}_i(N_i, D_i) = d_i(N_i - 1) + c_i D_i. \tag{2.32}$$

By merging the profits from (2.28) and (2.32), we can find the total profit as:

$$\pi_i(N_i, D_i) = \mathcal{P}_i(N_i) - \mathcal{N}_i(N_i, D_i) = \frac{(N_i P - \beta_i)^2}{4N_i^2 \gamma_i} - d_i(N_i - 1) - c_i D_i. \tag{2.33}$$

We call this expression the Component Profit Model. In this expression, the parameters P, β_i, γ_i, d_i, and c_i are fixed values that come with the model and do not change. There are, however, two variables: N_i and D_i. D_i is the number of direct connections a firm makes, and N_i is the total number of firms in the component. In essence, both N_i and D_i are derived from the network structure. We can either write the profit in terms of network properties as $\pi_i(N_i, D_i)$ or in terms of the network directly as $\pi_i(G)$. These two expressions are used interchangeably in this thesis and:

$$\pi_i(G) \equiv \pi_i(N_i, D_i). \tag{2.34}$$

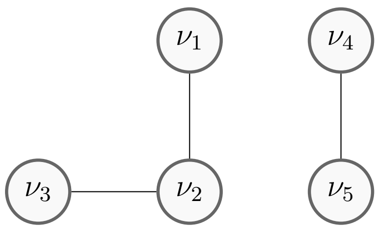
Another implication of this these equivalent expressions is that a firm that tries to increase its profit at step 2 may only do so by altering its network structure.

2.5 Stability with Component Profit Model

In Section 2.3, we discuss the multiple connectivity approach and stability by considering a general profit function. In this section, we consider the stability for networks when the component profit we describe in Section 2.4 model is applied.

2.5.1 MC Stability for a Firm

In this section, we consider the connection choices of a firm in an example system given in Figure 2.8 with $J = 5$ firms.

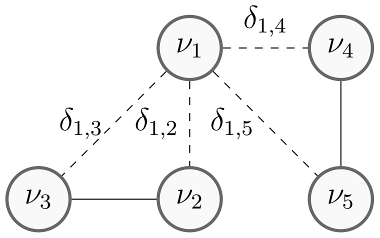


$$\equiv G = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}, D = \begin{bmatrix} 1 \\ 2 \\ 1 \\ 1 \\ 1 \end{bmatrix}, N = \begin{bmatrix} 3 \\ 3 \\ 3 \\ 2 \\ 2 \end{bmatrix}$$

Figure 2.8: Example system with five firms and two components.

In order to assess stability, we need to examine whether firms are able to propose a network that increases their profit that the other relevant firms would agree on forming. For this system, we need to look at five moves, one for each player.

Let us consider the move for firm 1. With four other firms, firm 1 may propose $2^4 = 16$ different networks, one of which is identical to G while others are different in i^{th} row and column. To capture this, we can define four choice variables $\delta_{i,k} \in \{0, 1\}$, which represents the choice of firm $i \in 1 \ldots J$ to connect with firm $k \in 1 \ldots J$ and $k \neq i$. Let us further define the function that alters G as $\mathcal{G}_i(\delta_i, G)$. This allows us to write the resultant network as a product of choice given in Figure 2.9.



$$\equiv \mathcal{G}_1(\delta_1, G) := \begin{bmatrix} 0 & \delta_{1,2} & \delta_{1,3} & \delta_{1,4} & \delta_{1,5} \\ \delta_{1,2} & 0 & 1 & 0 & 0 \\ \delta_{1,3} & 1 & 0 & 0 & 0 \\ \delta_{1,4} & 0 & 0 & 0 & 1 \\ \delta_{1,5} & 0 & 0 & 1 & 0 \end{bmatrix}$$

Figure 2.9: Possible connection choices for firm 1.

It can be seen that $\mathcal{G}_1(1, 0, 0, 0, G) = G$. Player 1 makes its choice as δ_1 so that its profit, which depends on N_1, and D_1, which in turn depends on G, is maximised. The optimisation problem for firm i connecting with firms $k \in 1 \ldots J$ and $k \neq i$ can be written as:

$$\begin{aligned} \max_{\delta_i} \ & \pi_i(\mathcal{G}_i(\delta_i, G)), \\ s.t. \ & \mathcal{R}_k(\delta_i, G)\delta_{i,k} \geq 0 \quad \forall k, \\ & \delta_{i,k} \in \{0, 1\} \quad \forall k. \end{aligned} \tag{2.35}$$

Let us assume $\hat{\delta}_i$ is the optimal solution for player i to the mixed integer optimisation problem defined in (2.35). If $\pi_i(\mathcal{G}_i(\hat{\delta}_i, G)) \leq \pi_i(G)$, the player has no incentive to alter the network when the network is already G.

2.5.2 MC Stability for a Network

In our system with five firms, finding the stability conditions requires us to solve five individual mixed integer problems and compare the results with the firms' profits emanating from the network G. After solving the optimisation problems, we can write five conditions to fulfil in order to demonstrate the stability of the system. First, each firm i's profits corresponding to player i's network choices in terms of D_i and

N_i are represented.

For each firm, $i \in \{1, \dots, J\}$ the network choice vector is $\delta_i \in \{0,1\}^{J-1}$. First, we convert the connection matrix as a result of choice ($\mathcal{G}_i(\delta_{i,k}, G)$) to the vectors of component size and direct connection number (N, D), which are the inputs of the profit function. Ideally, we'd like to represent $N^{(i)}(\delta_i)$ and $D^{(i)}(\delta_i)$ as parametric vectors. Doing this with a generalised method is discussed in detail later, but for now, we can simply inspect the system.

First, we see that the number of direct connections is the sum of $\delta_{1,k}$ and changing a single connection alters the number of direct connections by 1.

In terms of the component size, however, we need to consider that whether the firm is connected only to 2, only to 3 or both, its component size will only increase by 2. The same is also true for 4 and 5. To capture this, we use the sign function:

$$\text{sgn}(x) = \begin{cases} 1, & \text{if } x > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{2.36}$$

Then, when we write sgn($\delta_{1,2} + \delta_{1,3}$), which yields 1 if at least one of the connections is one, and 0 if neither is.

Finally, we need to consider that some firms would be connected indirectly due to connection choices. For the particular case of firm 2, its indirect connection to 4 and 5 can be represented as:

$$N_2 = 2 + \text{sgn}(\delta_{1,2} + \delta_{1,3})(1 + 2\text{sgn}(\delta_{1,4} + \delta_{1,5})). \tag{2.37}$$

Here, firm 2 has the component size of 2 as given since its connection to 3 is already included in the prior network. Then, if there is no connection between 1-2 and 1-3, whether one is connected to others is irrelevant. But if it is, the component size includes the secondary connections.

With all these considerations, we can write:

$$D^{(1)}(\delta_1) = D(\delta_{1,2}, \delta_{1,3}, \delta_{1,4}, \delta_{1,5}) = \begin{bmatrix} \delta_{1,2} + \delta_{1,3} + \delta_{1,4} + \delta_{1,5} \\ \delta_{1,2} + 1 \\ \delta_{1,3} + 1 \\ \delta_{1,4} + 1 \\ \delta_{1,5} + 1 \end{bmatrix}, \tag{2.38}$$

$$N^{(1)}(\delta_1) = N(\delta_{1,2}, \delta_{1,3}, \delta_{1,4}, \delta_{1,5}) = \begin{bmatrix} 1 + 2\text{sgn}(\delta_{1,2} + \delta_{1,3}) + 2\text{sgn}(\delta_{1,4} + \delta_{1,5}) \\ 2 + \text{sgn}(\delta_{1,2} + \delta_{1,3})(1 + 2\text{sgn}(\delta_{1,4} + \delta_{1,5})) \\ 2 + \text{sgn}(\delta_{1,2} + \delta_{1,3})(1 + 2\text{sgn}(\delta_{1,4} + \delta_{1,5})) \\ 2 + \text{sgn}(\delta_{1,4} + \delta_{1,5})(1 + 2\text{sgn}(\delta_{1,2} + \delta_{1,3})) \\ 2 + \text{sgn}(\delta_{1,4} + \delta_{1,5})(1 + 2\text{sgn}(\delta_{1,2} + \delta_{1,3})) \end{bmatrix}. \tag{2.39}$$

The sign function captures the notion that when either 2, 3, or both are connected (similarly for 4 and 5), the resultant component size does not change. In practice, when $D' < D$, $\pi(N, D) > \pi(N, D')$ so, we expect a player to never propose both connections and opt for the single connection to a component that offers larger profit while is agreeable to the other party. Then, we can restate the profit difference for firm k as:

$$\mathcal{R}_k^{(1)}(\delta_1) = \pi_k\big(N_k^{(1)}(\delta_1), D_k^{(1)}(\delta_1)\big) - \pi_k\big(N_k^{(1)}(0), D_k^{(1)}(0)\big), \tag{2.40}$$

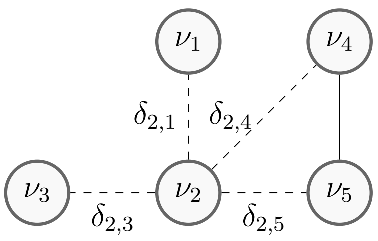
where $N_k^{(1)}(0)$ and $D_k^{(1)}(0)$ are the component size and the number of direct connections when $\delta_{1,k} = 0$ for all $k \in 2 \dots 5$, and our optimisation problem as:

$$\begin{aligned} &\max_{\delta_1} \ \pi_1(N_1^{(1)}(\delta_1), D_1^{(1)}(\delta_1)), \\ &s.t. \ \mathcal{R}_k^{(1)}(\delta_1)\delta_{1,k} \geq 0 \quad \forall k, \\ &\delta_{1,k} \in \{0, 1\} \quad \forall k. \end{aligned} \tag{2.41}$$

The solutions to such a problem can be found numerically using an optimisation technique such as exhaustive searching, branch and bound, or cutting plane methods [16]. Even with a strictly concave profit function, the solution to a mixed integer problem may not be unique. However, for the purposes of stability, we can define $\hat{\delta}_i$ as a maximiser to this problem and only need to check if:

$$\pi_1(N(\hat{\delta}_i), D(\hat{\delta}_i)) \leq \pi_1(G). \tag{2.42}$$

If $\hat{\delta}_i$ cannot increase profit compared to the prior matrix G, firm i would not alter any connections.



$$\equiv \mathcal{G}_2(\delta_2) := \begin{bmatrix} 0 & \delta_{2,1} & 0 & 0 & 0 \\ \delta_{2,1} & 0 & \delta_{2,3} & \delta_{2,4} & \delta_{2,5} \\ 0 & \delta_{2,3} & 0 & 0 & 0 \\ 0 & \delta_{2,4} & 0 & 0 & 1 \\ 0 & \delta_{2,5} & 0 & 1 & 0 \end{bmatrix}$$

Figure 2.10: Possible connection choices for firm 2.

Next, we consider how firm 2 may alter the network, which is given in Figure 2.10 and the parameters can be found as:

$$D^{(2)}(\delta_2) = \begin{bmatrix} \delta_{2,1} \\ \delta_{2,1} + \delta_{2,3} + \delta_{2,4} + \delta_{2,5} \\ \delta_{2,3} \\ \delta_{2,4} + 1 \\ \delta_{2,5} + 1 \end{bmatrix}, \tag{2.43a}$$

$$N^{(2)}(\delta_2) = \begin{bmatrix} 1 + \delta_{2,1}(1 + \delta_{2,3} + 2\text{sgn}(\delta_{2,4} + \delta_{2,5})) \\ 1 + \delta_{2,1} + \delta_{2,3} + 2\text{sgn}(\delta_{2,4} + \delta_{2,5}) \\ 1 + \delta_{2,3}(1 + \delta_{2,1} + 2\text{sgn}(\delta_{2,4} + \delta_{2,5})) \\ 2 + \text{sgn}(\delta_{2,4} + \delta_{2,5})(1 + \delta_{2,3} + \delta_{2,1}) \\ 2 + \text{sgn}(\delta_{2,4} + \delta_{2,5})(1 + \delta_{2,3} + \delta_{2,1}) \end{bmatrix}. \tag{2.43b}$$

These values can be used to write:

$$\mathcal{R}_k^{(2)}(\delta_2) = \pi_k\big(N_k^{(2)}(\delta_2), D_k^{(2)}(\delta_2)\big) - \pi_k\big(N_k^{(2)}(0), D_k^{(2)}(0)\big), \tag{2.44}$$

and

$$\begin{aligned} &\max_{\delta_2}\ \pi_2(N_2^{(2)}(\delta_2), D_2^{(2)}(\delta_2)), \\ &s.t.\ \mathcal{R}_k^{(2)}(\delta_2)\delta_{2,k} \geq 0 \quad \forall k, \\ &\delta_{2,k} \in \{0,1\} \quad \forall k. \end{aligned} \tag{2.45}$$

The solution to the underlying optimization problem, $\hat{\delta}_2$, generates the condition, $\pi_2(N(\hat{\delta}_2), D(\hat{\delta}_2)) < \pi_2(N, D)$. For the other three, the conditions are obtained similarly to the first two firms, so we can directly skip to the conclusion and write:

$$\pi_i(N_i^{(i)}(\hat{\delta}_i), D_i^{(i)}(\hat{\delta}_i)) < \pi_i(N_i, D_i) \quad \forall i = 1, 2, \dots 5. \tag{2.46}$$

If this condition does not hold, one of the firms has an acceptable alternative to the system G and would attempt to alter it as their move. If not, this system remains unchanged after playing the game. Thus, such a system would be stable.

2.5.3 Forest Condition

After making a stability inquiry for a particular system, we can now start making more general claims. Our first claim is regarding the shapes of possible stable networks. We recall that the component profit model described in Section **??** is:

$$\pi_i(G) \equiv \pi_i(N_i, D_i) = \frac{(PN_i - \beta_i)^2}{4\gamma_i N_i^2} - c_i D_i - d_i(N_i - 1). \tag{2.47}$$

Theorem 2. *A network with the component profit model is not MC or PW-stable if a connection can be removed while not reducing the component size of any firms.*

Proof. We assume one party of the connection in question is firm i. If we denote the network parameters for firm i with and without this connection as N_i, D_i and N_i', D_i', we can also write the relation:

$$N_i = N_i', \ \ D_i' = D_i - 1. \tag{2.48}$$

A difference in D_i without changing N_i creates a simple difference in the profit, which we can show as:

$$\pi_i(N_i, D_i - 1) - \pi_i(N_i, D_i) = c_i. \tag{2.49}$$

Since c_i is positive, so is the difference in profit for severing this connection.

Then, for the PW approach, the connection would be severed without further consideration. For the MC approach, the network firm i proposes only alters parameters for the party it is breaking from, which firm i is not obliged to consider. Therefore, if such a connection exists in a network, one of the parties would sever it, and such a system would not be stable. □

When the networks with multiple paths between two firms (also called cycles) are eliminated, the only remaining networks are forests as described in Section 2.2. This means with the component profit model, if there exists any stable network, it must be a forest (because non-forest networks cannot be stable).

Corollary 2.1 (Forest Condition)**.** *Forests are the only possible MC or PW stable networks with the component profit model.*

Neither Theorem 2 nor its corollary tells us that a stable network exists nor the exact topology it would take. It only tells us that a non-forest network cannot be stable.

2.5.4 The Optimal Component Size

The next thing we want to know is if there exists a size of component where firms would generate optimal profit. However, there are two problems we face.

- Component size may only take integer values.
- Reachable component sizes for firms depend on the prior network.

Former means that even if we find an optimal value for N_i, that value may not correspond to a possible component size, and there may be several N_i that provide the largest possible profit. Latter means that a firm may not be incentivised to alter the network even when they can improve their profit with a different N_i because there are no moves for them to reach it. Regardless, we can substitute discrete N and D with their continuous equivalents $n \in \mathbb{R}$ and $m \in \mathbb{R}$ as:

$$\pi_i(n, m) = \frac{(Pn - \beta_i)^2}{4\gamma_i n^2} - c_i m - d_i(n - 1), \tag{2.50}$$

and try to find expressions for the optimal continuous component size using the partial derivatives:

$$\frac{\partial \pi_i(n,m)}{\partial n} = \frac{P\beta_i}{2\gamma_i}\frac{1}{n^2} + \frac{-\beta_i^2}{2\gamma_i}\frac{1}{n^3} - d_i, \tag{2.51a}$$

$$\frac{\partial \pi_i(n,m)}{\partial m} = -c_i, \tag{2.51b}$$

$$\frac{\partial^2 \pi_i(n,m)}{\partial n^2} = \frac{-P\beta_i}{\gamma_i}\frac{1}{n^3} + \frac{3\beta_i^2}{2\gamma_i}\frac{1}{n^4}, \tag{2.51c}$$

$$\frac{\partial^2 \pi_i(n,m)}{\partial m^2} = \frac{\partial^2 \pi_i(n,m)}{\partial n \partial m} = \frac{\partial^2 \pi_i(n,m)}{\partial m \partial n} = 0. \tag{2.51d}$$

Then we can write the hessian matrix as:

$$H = \begin{bmatrix} \frac{-P\beta_i}{\gamma_i}\frac{1}{n^3} + \frac{3\beta_i^2}{2\gamma_i}\frac{1}{n^4} & 0 \\ 0 & 0 \end{bmatrix}. \tag{2.52}$$

One of the eigenvalues of the Hessian Matrix is zero. Therefore there exists no n, m pair that yields optimal profit. This is consistent with our prior finding that showed any profit can be increased by reducing m (or D_i) while keeping n (or N_i) constant. However, it may be possible for us to find optimal n for every constant m by showing:

$$0 > \frac{\partial^2 \pi_i(n,m)}{\partial n^2} = \frac{-P\beta_i}{\gamma_i}\frac{1}{n^3} + \frac{3\beta_i^2}{2\gamma_i}\frac{1}{n^4} = \frac{\beta_i}{2\gamma_i n^4}(3\beta_i - 2Pn). \tag{2.53}$$

Clearly, n^4, β_i, and γ_i are always positive. This means we only need to check:

$$0 > 1.5\beta_i - Pn. \tag{2.54}$$

It is previously established that $P > \beta_i$. Hence, for any $n > 1.5$, the profit is concave for a constant number of direct connections. For $n < 1.5$, however, it is convex. Moreover, it is not possible to keep the direct number of connections constant when $n = 1$, since $N_i = 1$ if and only if $D_i = 0$ for firm i. Thus, we can say that there exists an optimal component size for every firm that has at least one connection. We can find the optimal size of n by solving:

$$\frac{P\beta_i}{2\gamma_i}\frac{1}{n^2} + \frac{-\beta_i^2}{2\gamma_i}\frac{1}{n^3} - d_i = 0. \tag{2.55}$$

This expression does not depend on m. Therefore, we can further claim that the optimal component size is independent of the number of direct connections if those two values are independent. This expression is a cubic function, and finding an exact solution is difficult. Since we know there is a single solution within the boundary $n > 1.5$, we can find it with numeric techniques, such as picking a sufficiently high n and using bisection between it and $n = 1.5$.

2.5.5 Ceiling Condition

We know from Section 2.5.3 that forests are the only stable networks with our profit. Section 2.5.4 also tells us that there exists an optimal size of network for every firm. We can combine these two notions to write the ceiling condition.

Theorem 3. *Let $\widehat{K}_i$ be the optimal integer component size for firm i. A component of N firms cannot be MC stable if $\widehat{K}_i < K$ for all i.*

Proof. Assume that the component size is indeed larger than $\widehat{K}_i$ for all i. When $K > \widehat{K}_i$, the profit of firms can be improved by decreasing their component size by one for all firms. If the component is a tree, there must be at least one firm which is only connected to the rest of the component with a single connection (a leaf node). The firm connected to the leaf firm can propose an alteration by severing the leaf to decrease the component size by one, and all its other connections would consent as their profits also increase by this new network. Thus, it would not be stable. If the component is not a tree, then it is not stable. Hence, a stable network must not be larger than all the optimal component sizes of its member firms. □

We might then be inclined to assume $\widehat{K}_i < K$ only for the non-leaf nodes would be equivalent to the theorem. However, one of the other connections that must consent to the new network may also be a leaf, and severing two leaves may be less profitable for firms. Thus, such a claim cannot be made.

However, when we consider pairwise stability, we can indeed make a stronger claim:

Theorem 4. *A component of K firms cannot be PW stable if $\widehat{K}_i < K$ for any firm connected to a firm with a single connection.*

Proof. The proof is similar to the proof of the previous theorem, but this time, since the relevant firm does not require the consent of its direct connections, it can simply sever a leaf to increase its own profit. $\square$

The ceiling condition implies that even when J is very large, we expect a limit to the component size, and stable networks may only include multiple components no larger than $\widehat{K}$.

2.5.6 Firms with Equal Parameters

Our final assertion in this section is about the particular case where all firms have the same profit function parameters (β, γ, c, d), which can be shown as:

$$\pi_i(N_i, D_i) = \pi(N_i, D_i) = \frac{(PN_i - \beta)^2}{4\gamma N_i^2} - cD_i - d(N_i - 1) \qquad \forall\, i. \tag{2.56}$$

Theorem 5. *If network G is MC or PW stable with firms with the same profit function parameters, isomorphic systems to G are also MC or PW stable, respectively.*

Proof. Isomorphism is introduced in Section 2.2.3 as one network being obtainable from the other by only changing the labelling. We can call two firm networks isomorphic if we can obtain one system from the other by changing which node represents which firm.

In Section 2.2.3, we further state that N and D vectors between two isomorph graphs only differ by their index. Then, if two networks are isomorphs and the profit functions are the same for all firms, the profits that firms generate would also only differ by index.

For an MC stable network, we know that all firms solve the optimisation problems that arise from their networks and cannot find an alteration that improves their profit. For a PW stable network, no firm can propose improving the existing network by severing or making a connection. In a system with equal parameters, either of these situations is independent of the firm that makes them and only depends on N and D vectors. Therefore, in every isomorph of G, there exist the same choices for firms which are numbered differently. By the stability results from all the firms' choices, we can say that if a system of equal parameters and network G is stable, isomorphic systems are also stable. $\square$

Theorem 5 further means that when we consider firms with equal parameters, it is not necessary to check every single forest for stability. Instead, checking unique isomorph forests is enough.

2.6 Example of a Stable Network

While we are able to make some assertions regarding how a stable network may be, without further information on the profit functions of firms and the relevant network, it is still difficult to finalise the discussion on stability. Therefore, in this section, we consider an example system with three firms with $P = 10$ and firm parameters given in Table 2.7.

Table 2.7: Example firm coefficients.

firm	β_i	γ_i	c_i	d_i
1	5	2	2	0.1
2	4	2	1	0.1
3	6	2	1	0.1

2.6.1 The Unconnected Network

We start by considering a network without connections, which can be represented by a matrix of zeros G_0. This leads to the function of possible connections for firm i:

$$\mathcal{G}_1(\delta_1, G_0) = \begin{bmatrix} 0 & \delta_{1,2} & \delta_{1,3} \\ \delta_{1,2} & 0 & 0 \\ \delta_{1,3} & 0 & 0 \end{bmatrix}. \tag{2.57}$$

This allows us to write:

$$N^{(1)}(\delta_1) = \begin{bmatrix} 1 + \delta_{1,2} + \delta_{1,3} \\ 1 + \delta_{1,2}(1 + \delta_{1,3}) \\ 1 + \delta_{1,3}(1 + \delta_{1,2}) \end{bmatrix}, \; D^{(1)}(\delta_1) = \begin{bmatrix} \delta_{1,2} + \delta_{1,3} \\ \delta_{1,2} \\ \delta_{1,3} \end{bmatrix}. \tag{2.58}$$

We can substitute (2.58) into (2.33), and find the profit functions that depend on δ_1 as:

$$\pi_1(\delta_1) = \frac{(5 + 10\delta_{1,2} + 10\delta_{1,3})^2}{8(1 + \delta_{1,2} + \delta_{1,3})^2} - 0.1(\delta_{1,2} + \delta_{1,3}) - 2(\delta_{1,2} + \delta_{1,3}), \tag{2.59a}$$

$$\pi_2(\delta_1) = \frac{(6 + 10\delta_{1,2}(1 + \delta_{1,3}))^2}{8(1 + \delta_{1,2}(1 + \delta_{1,3}))^2} - 0.1(\delta_{1,2}(1 + \delta_{1,3})) - \delta_{1,2}, \tag{2.59b}$$

$$\pi_3(\delta_1) = \frac{(4 + 10\delta_{1,3}(1 + \delta_{1,2}))^2}{8(1 + \delta_{1,3}(1 + \delta_{1,2}))^2} - 0.1(\delta_{1,3}(1 + \delta_{1,2})) - \delta_{1,3}. \tag{2.59c}$$

Hence, we can write the problem directly in terms of $\delta_1 \in \{0,1\}^2$ and $k \in \{2,3\}$ as:

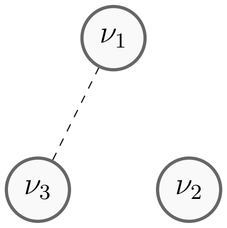
$$\begin{aligned} &\max_{\delta_1} \pi_1(\delta_1), \\ &s.t.\ \delta_{1,k}(\pi_k(\delta_1) - \pi_k(0)) \geq 0, \\ &\quad \delta_{1,k} \in \{0,1\}. \end{aligned} \tag{2.60}$$

The vector δ_1 has two elements, leading to four possible combinations. On such a small scale, we can simply consider all possible connections to find the one that yields the highest profit. This is presented in Table 2.8.

Table 2.8: Firm profits with respect to choices of firm 1.

$\delta_{1,2}$	$\delta_{1,3}$	π_1	π_2	π_3
0	0	3.125	4.5	2.0
1	0	4.931	6.9	2.0
0	1	4.931	4.5	5.025
1	1	4.481	8.19	6.8

The values show us that the matrix of zeros is not stable under MCG. In fact, any connection a firm can offer will increase its profit. We further see that making both connections is not as profitable as making a single connection, but the firm's profit is the same between the two options. Moreover, we see that both firms 2 and 3 consent to making that single connection as $5.025 > 2$ and $6.9 > 4.5$. For this example, let us say that firm 1 is able to alter the network by connecting with firm 3. Thus we continue with G_1 as given in Figure 2.11:



$$\equiv G_1 := \begin{bmatrix} 0 & 0 & \mathbf{1} \\ 0 & 0 & 0 \\ \mathbf{1} & 0 & 0 \end{bmatrix}$$

Figure 2.11: Best connection choice for firm 1.

2.6.2 A Network with a Single Connection

The given firms have different parameters. Therefore, it is possible to analyse the stability of all three different pairs. But since we know G_1 would be the best choice for firm 1, let us continue examining it with another player, firm 2. We start by writing:

$$\mathcal{G}_2(\delta_2, G_1) = \begin{bmatrix} 0 & \delta_{2,1} & 1 \\ \delta_{2,1} & 0 & \delta_{2,3} \\ 1 & \delta_{2,3} & 0 \end{bmatrix}, \tag{2.61}$$

which allows us to write:

$$N^{(2)}(\delta_2) = \begin{bmatrix} 2 + \text{sgn}(\delta_{2,1} + \delta_{2,3}) \\ 1 + 2\text{sgn}(\delta_{2,1} + \delta_{2,3}) \\ 2 + \text{sgn}(\delta_{2,1} + \delta_{2,3}) \end{bmatrix}, \; D^{(2)}(\delta_2) = \begin{bmatrix} 1 + \delta_{2,1} \\ \delta_{2,1} + \delta_{2,3} \\ 1 + \delta_{2,3} \end{bmatrix}, \tag{2.62}$$

and:

$$\pi_1(\delta_2) = \frac{(15 + 10\text{sgn}(\delta_{2,1} + \delta_{2,3}))^2}{8(2 + \text{sgn}(\delta_{2,1} + \delta_{2,3}))^2} - 0.1(1 + \text{sgn}(\delta_{2,1} + \delta_{2,3})) - 2(1 + \delta_{2,1}), \tag{2.63a}$$

$$\pi_2(\delta_2) = \frac{(6 + 20\text{sgn}(\delta_{2,1} + \delta_{2,3}))^2}{8(1 + 2\text{sgn}(\delta_{2,1} + \delta_{2,3}))^2} - 0.2\text{sgn}(\delta_{2,1} + \delta_{2,3}) - (\delta_{2,1} + \delta_{2,3}), \tag{2.63b}$$

$$\pi_3(\delta_2) = \frac{(14 + 10\text{sgn}(\delta_{2,1} + \delta_{2,3}))^2}{8(2 + \text{sgn}(\delta_{2,1} + \delta_{2,3}))^2} - 0.1(1 + \text{sgn}(\delta_{2,1} + \delta_{2,3})) - (1 + \delta_{2,3}). \tag{2.63c}$$

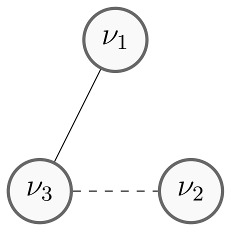
As done before, we use the sign function to express whether a connection from 2 to 3 or 1 is established. The profits from these choices can be seen in Table 2.9.

We see that while it is not profitable for firm 1 to alter G_1, firm 2 would benefit from it. Furthermore, firm 2 would prefer to make a single connection instead of both.

Table 2.9: Firm profits with respect to choices of firm 2.

$\delta_{2,1}$	$\delta_{2,3}$	π_1	π_2	π_3
0	0	4.931	4.5	5.025
1	0	4.481	8.19	6.8
0	1	6.481	8.19	5.8
1	1	4.481	7.19	5.8

Moreover, the profit of firm 1 reduces with any possible connection. Hence, it would not consent to any connection. However, for firm 3, we have $5.8 > 5.025$, so in order to maximise its profit, firm 2 would propose forming a network with a connection to firm 3 while ignoring the preference of firm 1 as it is not to be connected. The resultant network is shown in Figure 2.12.



$$\equiv G_2 := \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & \mathbf{1} \\ 1 & \mathbf{1} & 0 \end{bmatrix}$$

Figure 2.12: Best connection for firm 2.

2.6.3 A Stable Configuration

We know that G_0 and G_1 are not stable. So now we consider G_2. The choice firm 3 appears somewhat different from those before, as it is already connected to both firms. If any possible alternative is less profitable than its current profit of 5.8, it may choose not to alter the network. The network affected by the choice variables can be given as:

$$\mathcal{G}_3(\delta_3, G_2) = \begin{bmatrix} 0 & 0 & \delta_{3,1} \\ 0 & 0 & \delta_{3,2} \\ \delta_{3,1} & \delta_{3,2} & 0 \end{bmatrix}, \tag{2.64}$$

and parameters as:

$$N^{(3)}(\delta_3) = \begin{bmatrix} 1 + \delta_{3,1}(1 + \delta_{3,2}) \\ 1 + \delta_{3,2}(1 + \delta_{3,1}) \\ 1 + \delta_{3,1} + \delta_{3,2} \end{bmatrix}, \; D^{(3)}(\delta_3) = \begin{bmatrix} \delta_{3,1} \\ \delta_{3,2} \\ \delta_{3,1} + \delta_{3,2} \end{bmatrix}. \tag{2.65}$$

We can substitute the relevant values to the general profit expression (2.33) and find the profit functions:

$$\pi_1(\delta_3) = \frac{(5 + 10\delta_{3,1}(\delta_{3,2} + 1))^2}{8(1 + \delta_{3,1}(\delta_{3,2} + 1))^2} - 0.1(\delta_{3,1}(\delta_{3,2} + 1)) - 2\delta_{3,1}, \tag{2.66a}$$

$$\pi_2(\delta_3) = \frac{(6 + 10\delta_{3,2}(\delta_{3,1} + 1))^2}{8(1 + \delta_{3,2}(\delta_{3,1} + 1))^2} - 0.1(\delta_{3,2}(\delta_{3,1} + 1)) - \delta_{3,2}, \tag{2.66b}$$

$$\pi_3(\delta_3) = \frac{(4 + 10\delta_{3,1} + 10\delta_{3,2})^2}{8(1 + \delta_{3,1} + \delta_{3,2})^2} - 0.1(\delta_{3,1} + \delta_{3,2}) - \delta_{3,1} - \delta_{3,2}. \tag{2.66c}$$

Table 2.10: Firm profits with respect to choices of firm 3.

$\delta_{3,1}$	$\delta_{3,2}$	π_1	π_2	π_3
0	0	3.125	4.5	2.0
1	0	4.931	4.5	5.025
0	1	3.125	6.9	5.025
1	1	6.481	8.189	5.8

The profits from choices are found in Table 2.10, and we can observe that the most profitable choice for the firm is to preserve the existing network structure. We moreover know that firm 2 already considers this to be a network to not deviate from. So, in order to assess stability, we only need to make sure firm 1, too, is not incentivised to deviate from this system.

We start by noting that firms 2 and 3 are in a component. Hence, we write:

$$\mathcal{G}_1(\delta_1, G_3) = \begin{bmatrix} 0 & \delta_{1,2} & \delta_{1,3} \\ \delta_{1,2} & 0 & 1 \\ \delta_{1,3} & 1 & 0 \end{bmatrix}, \tag{2.67}$$

and

$$N^{(4)}(\delta_1) = \begin{bmatrix} 1 + 2\text{sgn}(\delta_{1,2} + \delta_{1,3}) \\ 2 + \text{sgn}(\delta_{1,2} + \delta_{1,3}) \\ 2 + \text{sgn}(\delta_{1,3} + \delta_{1,2}) \end{bmatrix}, D^{(4)}(\delta_1) = \begin{bmatrix} \delta_{1,2} + \delta_{1,3} \\ 1 + \delta_{1,2} \\ 1 + \delta_{1,3} \end{bmatrix}. \tag{2.68}$$

The profit functions from these values can be written as:

$$\pi_1(\delta_1) = \frac{(5 + 20\text{sgn}(\delta_{1,2} + \delta_{1,3}))^2}{8(1 + 2\text{sgn}(\delta_{1,2} + \delta_{1,3}))^2} - 0.2\text{sgn}(\delta_{1,2} + \delta_{1,3}) - 2(\delta_{1,2} + \delta_{1,3}), \tag{2.69a}$$

$$\pi_2(\delta_1) = \frac{(16 + 10\text{sgn}(\delta_{1,2} + \delta_{1,3}))^2}{8(2 + \text{sgn}(\delta_{1,2} + \delta_{1,3}))^2} - 0.1(1 + \text{sgn}(\delta_{1,2} + \delta_{1,3})) - (1 + \delta_{1,2}), \tag{2.69b}$$

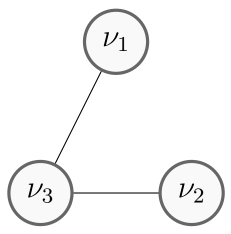
$$\pi_3(\delta_1) = \frac{(14 + 10\text{sgn}(\delta_{1,3} + \delta_{1,2}))^2}{8(2 + \text{sgn}(\delta_{1,3} + \delta_{1,2}))^2} - 0.1(1 + \text{sgn}(\delta_{1,3} + \delta_{1,2})) - (1 + \delta_{1,3}), \tag{2.69c}$$

and from these, we can compute the profits as given in Table 2.11.

Table 2.11: Firm profits with respect to choices of firm 1.

$\delta_{1,2}$	$\delta_{1,3}$	π_1	π_2	π_3
0	0	3.125	6.9	5.025
1	0	6.481	7.189	6.8
0	1	6.481	8.189	5.8
1	1	4.481	7.189	5.8

Firm 1 starts playing with a profit of 6.481, which it is not able to improve. Hence, it is clear that firm 1 would also choose to preserve the existing network. Thus, we can say that no firm would prefer to alter the network structure. Thus, for conditions described at the beginning of this example, firms that maximise their profit would be stable if the network described in Figure 2.13 is formed.



$$\equiv \widehat{G} := \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

Figure 2.13: Stable network for this example.

We thus conclude our examples regarding how the stability of networks would be examined with the MC approach. In the next section, we present a summary of this chapter.

2.7 Summary of the Model

We start this chapter with firms that compete in the same market over the cost of a single good they produce. Profit-maximising firms naturally produce the optimal amount of goods with regard to their cost function. Hence, by relating the cost terms to network properties and assuming optimal production, we can express the firms' profit entirely on the properties obtained from their locations in a network. We use two network properties, component size and direct connections, which we express by vectors $N_i \in \{1, 2, \dots J\}^J$, and $D_i \in \{0, 1, \dots J-1\}^J$, respectively, for J firms. And we find the profit function:

$$\pi_i(N_i, D_i) = \frac{(N_i P - \beta_i)^2}{4N_i^2 \gamma_i} - d_i(N_i - 1) - c_i D_i. \tag{2.70}$$

In this expression, β_i and γ_i correspond to linear and quadratic production costs, respectively, and c_i and d_i are connection and component costs, respectively.

While we make the discussion of stability without referencing a particular game structure, it is possible to state the choice firms make to alter or preserve the network they are in as the Multiple Connectivity Game for J firms:

- Game is played with J number of firms connected by a symmetric adjacency matrix G.
- Each firm can choose to preserve the network or propose an alternative network by altering its direct connections.
- The firms that would be directly connected to the proposing firm must also consent to the new network, which requires them to be not worse off than the previous network.
- If no firm can profit by proposing an alternative to G, the system is stable.

Then, we can define the choice of forming a connection from i to k as $\delta_{i,k}$ and $\delta_i \in \{0, 1\}^{J-1}$ as the vector of all the choices of firm i at their turn. This allows us to write the difference in profit for firm k as:

$$\mathcal{R}_k(\delta_i, G) = \pi_k\big(\mathcal{G}_i(\delta_i, G)\big) - \pi_k\big(G\big), \tag{2.71}$$

and the optimisation problem to find the best move for a player $i \in \{1, \dots, J\}$ connecting to players $k \in \{1, \dots, J\}$ and $k \neq i$ as:

$$\begin{aligned} \max_{\delta_i} \ & \pi_i\big(\mathcal{G}_i(\delta_i, G)\big), \\ s.t. \ & \mathcal{R}_k(\delta_i, G)\delta_{i,k} \geq 0 \ \ \forall k, \\ & \delta_{i,k} \in \{0, 1\} \ \ \forall k. \end{aligned} \tag{2.72}$$

Our discussion in this chapter is mostly limited to how our problem is modelled and how we can assess stability for firms that are connected in a network. This is done by relying on simple examples that require simple computations. However, in order to solve more complex cases, it is clear that more advanced techniques need to be employed. In the next chapter, a general method to solve problems of this sort is described, and an algorithm to solve systems with more complex conditions is presented.

CHAPTER 3

A NUMERICAL APPROACH FOR THE MULTIPLE CONNECTIVITY MODEL

3.1 Introduction

Within this thesis we present a game that can be used to expand the understanding of network formation. To that end, Chapter 2 explains how our model of firms came to be. This explanation includes a way to tie the profit of firms to the network interactions and our novel game in which firms can alter all their connections as a single move to form a network structure in order to increase their profit. However, in Chapter 2, the discussion on how firms can find the optimal moves while playing this game remains on the surface level. How to solve the resultant optimization problem is not necessarily integral to our proposed model. Nevertheless, due to the novelty of the multi-choice approach, we consider it appropriate to discuss solvability of the problem. Therefore this chapter studies the question of finding the optimal move for players of a Multiple Connectivity Game in greater detail.

In order to solve our non-linear mixed integer problem (MINLP), we first need a way to relax integer constraints [20]. In Section 3.2, we propose an approximation method that we can use for these relaxations and our reasoning for following this approach. We follow this with a discussion on the concavity of the maximization problem in Section 3.3. Having found an equivalent problem for relaxed variables that also has a concave profit function, we are able to solve this problem with a tree-based algorithm [7]. In particular, we choose to modify branch and bound algorithm for its relative simplicity while solving our problem. A broader discussion on the algorithm and our modifications can be found in Section 3.4. Finally, we provide a procedure that finds MC stable systems in Section 3.5 to conclude our discussion on the numerical solution for the Multiple Connectivity Game.

3.2 Continuous Interpolations of the Network Properties

3.2.1 Motivation for the Interpolation

Within this thesis, we consider the effects of being in a network for J firms based on two variables. These are $N \in \{1, \dots, J\}^J$ and $D \in \{0, \dots, J-1\}^J$, which are component size and direct connection number vectors, respectively. We obtain these values by operating on the adjacency matrix and then use them to find the profits for firms. As discussed in Section 2.7, the best connections a firm can make with a prior network $G \in \{0,1\}^{J \times J}$ can be found by solving the following optimization problem:

$$\begin{aligned} \max_{\delta_i} \;\; & \pi_i\big(\mathcal{G}_i(\delta_i, G)\big), \\ s.t. \;\; & \mathcal{R}_k(\delta_i, G)\delta_{i,k} \geq 0 \quad \forall k, \\ & \delta_{i,k} \in \{0,1\} \quad \forall k. \end{aligned} \tag{3.1}$$

This problem is a non-linear mixed integer problem (sometimes called mixed integer non-linear programming problem or MINLP). The solution methods that yield deterministic solutions to such problems often require relaxing the integer parameters [20]. Relaxing an integer variable in this context refers to substituting it with a continuous variable and using continuous values between integers to approach the solution of the main problem. For our case the condition $\delta_{i,k} \in \{0,1\}$ would be relaxed to the condition $0 \leq \delta_{i,k} \leq 1$. However, due to the method we compute N, relaxing choice variables does not correspond to a smooth change in the profit. In Section 2.2.2, we show that the component size for firm a[1] is:

$$N_a = \sum_b \text{sgn}((G + \mathbf{I})^{J-1})_{a,b}. \tag{3.2}$$

[1] We substitute N_i with N_a to avoid clashing with δ_i.

We can use (3.2) to find the function for N in terms of the decision variables δ_i by substituting G with $\mathcal{G}_i$ as:

$$N_a(\delta_i, G) = \sum_b \operatorname{sgn}\big((\mathcal{G}_i(\delta_i, G) + \mathbf{I})^{J-1}\big)_{a,b}. \tag{3.3}$$

Due to the sign function, while $\delta_{i,k} = 0$ and $\delta_{i,k} = 1$ yield different N, the intermediate values for $\delta_{i,k}$ would be equal to N obtained from $\delta_{i,k} = 1$. This makes relaxation meaningless for δ_i. To address problems around relaxing integer variables that arise from non-linearities in the objective function, the Outer Approximation Method is proposed in the literature. The method converts the feasible space into a polyhedral representation [18]. For this study, we follow a similar approach and convert the component size function in (3.3) to a polyhedral representation to enable variable relaxation while solving the optimization problem.

3.2.2 Continuous Component Size Interpolation

While approximating intermediate values for component size N, our priority would be making sure that $N(\delta_i)$ (for simplicity, we write $N(\delta_i)$ instead of $N(\delta_i, G)$ in this Section) yields the correct results for the integer values of δ_i. Since we plan to use intermediate values as mere guides, we are content with using a linear approximation. To that end, our first step is writing linear change in N when just one $\delta_{i,k}$ changes and others are zero.

First, we write $N^0 = N(0)$ and define a matrix $M_{i,k} \in \{0, 1, 2, \dots, J-1\}^{J \times J-1}$, where rows represent the relevant firm and columns represent the difference in N values when a single connection is made to k by i. This matrix can be written in the form:

$$M = [N(1, 0, \dots) - N^0,\ N(0, 1, 0 \dots) - N^0, \dots, N(0, \dots, 0, 1) - N^0]. \tag{3.4}$$

Then, we can interpolate the intermediate values for the single connection to k ($\delta_{i,l} \neq 0$ only when $l = k$) case as:

$$N(\delta_i) = N^0 + M\delta_k. \tag{3.5}$$

For firm i, this expression is sufficient as its component size is directly influenced by every choice. Thus for firm i in particular, we can further note that $N_i^0 = 1$ since without any connections component size of the connecting firm is 1, and write:

$$N_i(\delta_i) = 1 + \sum_k M_{i,k}\delta_{i,k}. \tag{3.6}$$

With this expression, if some firms are in a component, their effect is captured multiple times, but for now, let us assume only a single connection is made to each component. We discuss making multiple connections to a component in Section 3.2.4.

For firms other than i, however, we must also note that the effect of other connections must first go through their connection to i. We can capture this phenomenon by writing:

$$N_a(\delta_i) = N_a^0 + \left(\sum_k M_{a,k}\delta_{i,k}\right)\left(\sum_k M_{i,k}\delta_{i,k} - N_a^0\left(\sum_k M_{a,k}\delta_{i,k}\right) + 1\right). \tag{3.7}$$

The first part of the multiplication is the effect of choice to get connected to firm i, which is then multiplied by the effect of every connection to firm i. Next, the effect connected to the component of k is subtracted, and finally, 1 represents the single increase in N due to being connected to firm i. We note that when we replace a with i in (3.7), we find (3.6).

3.2.3 Direct Connection Approximation

Our expression for direct connections of firm a from Section 2.2.2 is:

$$D_a(G) = \sum_b G_{a,b}. \tag{3.8}$$

The equivalent function that depends on choice variables can be written as:

$$D_a(\delta_{i,k}, G) = \sum_b \mathcal{G}_i(\delta_i, G)_{a,b}. \tag{3.9}$$

While the underlying variable is discrete by nature, similar to N, substituting it with a continuous variable directly leaves us with the linear approximated values for intermediate D. Thus we can use (3.9) in our relaxations.

3.2.4 Demonstration of the Continuous Approximations

For the demonstration of the proposed approach, we reconsider the example with three firms discussed in Section 2.6, where the decision matrix is given by:

$$\mathcal{G}_1(\delta_{1,2}, \delta_{1,3}, G_0) = \begin{bmatrix} 0 & \delta_{1,2} & \delta_{1,3} \\ \delta_{1,2} & 0 & 0 \\ \delta_{1,3} & 0 & 0 \end{bmatrix}, \tag{3.10}$$

and N is inspected to be:

$$N(\delta_1) = \begin{bmatrix} 1 + \delta_{1,2} + \delta_{1,3} \\ 1 + \delta_{1,2}(1 + \delta_{1,3}) \\ 1 + \delta_{1,3}(1 + \delta_{1,2}) \end{bmatrix}. \tag{3.11}$$

We follow the steps of interpolation method discussed in Section 3.2.2, and first find N^0 and N for two connections individually:

$$N^0 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, N(1,0) = \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix}, N(0,1) = \begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix}, \tag{3.12}$$

which gives us the matrix M:

$$M = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{3.13}$$

Using M from (3.13), we can write:

$$N(\delta_1)_1 = 1 + (\delta_{1,2} + \delta_{1,3}), \tag{3.14a}$$

$$N(\delta_1)_2 = 1 + (\delta_{1,2})(\delta_{1,2} + \delta_{1,3} - \delta_{1,2} + 1) = 1 + \delta_{1,2}(\delta_{1,3} + 1), \tag{3.14b}$$

$$N(\delta_1)_3 = 1 + (\delta_{1,3})(\delta_{1,2} + \delta_{1,3} - \delta_{1,3} + 1) = 1 + \delta_{1,3}(\delta_{1,2} + 1), \tag{3.14c}$$

and as a vector:

$$N(\delta_1) = \begin{bmatrix} 1 + (\delta_{1,2} + \delta_{1,3}) \\ 1 + \delta_{1,2}(\delta_{1,3} + 1) \\ 1 + \delta_{1,3}(\delta_{1,2} + 1) \end{bmatrix}, \tag{3.15}$$

which is identical to the N obtained in Section 2.6.1.

Next, we consider the second part of the example:

$$\mathcal{G}_2(\delta_{2,1}, \delta_{2,3}, G_1) = \begin{bmatrix} 0 & \delta_{2,1} & 1 \\ \delta_{2,1} & 0 & \delta_{2,3} \\ 1 & \delta_{2,3} & 0 \end{bmatrix}, \; N(\delta_2) = \begin{bmatrix} 2 + \text{sgn}(\delta_{2,1} + \delta_{2,3}) \\ 1 + 2\text{sgn}(\delta_{2,1} + \delta_{2,3}) \\ 2 + \text{sgn}(\delta_{2,1} + \delta_{2,3}) \end{bmatrix}. \tag{3.16}$$

Before applying the method we derived, we need to remove the sign function. First, we recall the theorem from Section 2.5.3 which asserts that a firm is not incentivised to make multiple connections to the same component. We can further argue that when a firm has multiple choices to connect to the same component, it would always prefer making a single connection. Establishing this further means that we can add the condition of a single connection to a component to the general optimization problem and thus make sure that within the sgn(), it is always either 1 or 0, which allows us to neglect it safely. This finding alters the inspected expression (3.16) for N to:

$$N(\delta_2) = \begin{bmatrix} 2 + (\delta_{2,1} + \delta_{2,3}) \\ 1 + 2(\delta_{2,1} + \delta_{2,3}) \\ 2 + (\delta_{2,1} + \delta_{2,3}) \end{bmatrix}, \tag{3.17}$$

Next we apply our method. We first look at N^0 and individual connections and assemble M:

$$N^0 = \begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix}, \; N(1,0) = \begin{bmatrix} 3 \\ 3 \\ 3 \end{bmatrix}, \; N(0,1) = \begin{bmatrix} 3 \\ 3 \\ 3 \end{bmatrix}, \; M = \begin{bmatrix} 1 & 1 \\ 2 & 2 \\ 1 & 1 \end{bmatrix}. \tag{3.18}$$

We use these to write:

$$N(\delta_2)_1 = 2 + (\delta_{2,1} + \delta_{2,3})(2\delta_{2,1} + 2\delta_{2,3} - 2(\delta_{2,1} + \delta_{2,3}) + 1) = 2 + (\delta_{2,1} + \delta_{2,3}), \tag{3.19a}$$

$$N(\delta_2)_2 = 1 + 2(\delta_{2,1} + \delta_{2,3}), \tag{3.19b}$$

$$N(\delta_2)_3 = 2 + (\delta_{2,1} + \delta_{2,3})(2\delta_{2,1} + 2\delta_{2,3} - 2(\delta_{2,1} + \delta_{2,3}) + 1) = 2 + (\delta_{2,1} + \delta_{2,3}), \tag{3.19c}$$

and as a vector:

$$N(\delta_2) = \begin{bmatrix} 2 + (\delta_{2,1} + \delta_{2,3}) \\ 1 + 2(\delta_{2,1} + \delta_{2,3}) \\ 2 + (\delta_{2,1} + \delta_{2,3}) \end{bmatrix}. \tag{3.20}$$

These expressions are identical to our derived expression (3.20). With our method, we can start from any network G and write an adequate approximation that can be used for solving optimization problems regarding firms' choices.

3.2.5 Multiple Connections to the Same Component

In the previous section, in order to remove the sign function, we assert that multiple connections to the same component would be prohibited by including a condition to the optimization problem. In order to do this, it is necessary to formulate these conditions in terms of δ_i and G.

We begin with the matrix used to compute N, which is $G+\mathbf{I}$ as given in Section 2.2.2. Since we are searching for components without firm i, we can assume it is not connected and write $G = \mathcal{G}_i(0, G)$. Then,

$$\operatorname{sgn}(\mathcal{G}_i(0, G)^{J-1} + \mathbf{I}), \tag{3.21}$$

gives us a matrix where components excluding i are completely connected. We can multiply that matrix with the choice vector δ_i to find the expression:

$$B(\delta_i, G)_j = \sum_k \operatorname{sgn}\big((\mathcal{G}_i(0, G) + \mathbf{I})^{J-1}\big)_{j,k}\delta_{i,k}. \tag{3.22}$$

If there are multiple connections to the same component, (3.22) is larger than 1. This can be illustrated when we look at the five firm examples from a single connection case:

$$G = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}. \tag{3.23}$$

If we assume firm 4 is making the connection choice, then we can write it as:

$$\mathcal{G}_4(\delta_4) = \begin{bmatrix} 0 & 1 & 0 & \delta_{4,1} & 0 \\ 1 & 0 & 1 & \delta_{4,2} & 0 \\ 0 & 1 & 0 & \delta_{4,3} & 0 \\ \delta_{4,1} & \delta_{4,2} & \delta_{4,3} & 0 & \delta_{4,5} \\ 0 & 0 & 0 & \delta_{4,5} & 0 \end{bmatrix}. \tag{3.24}$$

Here, there is a component of three firms and one unconnected firm. We can use the method described in Section 3.2.2 to write:

$$N(\delta_4) = \begin{bmatrix} 3 + (\delta_{4,1} + \delta_{4,2} + \delta_{4,3})(1 + \delta_{4,5}) \\ 3 + (\delta_{4,1} + \delta_{4,2} + \delta_{4,3})(1 + \delta_{4,5}) \\ 3 + (\delta_{4,1} + \delta_{4,2} + \delta_{4,3})(1 + \delta_{4,5}) \\ 1 + 3(\delta_{4,1} + \delta_{4,2} + \delta_{4,3}) + \delta_{4,5} \\ 1 + \delta_{4,5}(1 + (\delta_{4,1} + \delta_{4,2} + \delta_{4,3})) \end{bmatrix}. \tag{3.25}$$

It is clear that without the restriction of $\delta_{4,1} + \delta_{4,2} + \delta_{4,3} \leq 1$, $N(\delta_4)$ would reach values much higher than $J = 5$. Hence we need to apply the component condition. We start by substituting relevant values to (3.21):

$$\text{sgn}(\mathcal{G}_4(0, G)^{J-1} + \mathbf{I}) = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}^{J-1} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.26}$$

When we multiply this expression with δ_4, we obtain:

$$B(\delta_4)_1 = B(\delta_4)_2 = B(\delta_4)_3 = \delta_{4,1} + \delta_{4,2} + \delta_{4,3}, \tag{3.27a}$$

$$B(\delta_4)_5 = \delta_{4,5}. \tag{3.27b}$$

If these expressions are both ≤ 1, we can make sure that multiple connections are not made to the same component.

To summarise, we can write the general component condition as:

$$B(\delta_i, G)_j = \sum_k \text{sgn}\big((\mathcal{G}_i(0, G) + \mathbf{I})^{J-1}\big)_{j,k} \delta_{i,k} \leq 1 \quad \forall j. \tag{3.28}$$

3.2.6 Interpolated Objective Function

Having handled the difficulties in expressing the problem in terms of choice in a meaningful way, we can finally write the interpolated problem. First, we recall the problem we start with from Section 2.7:

$$\begin{aligned} \max_{\delta_i} \ & \pi_i(\mathcal{G}_i(\delta_{i,k}, G)), \\ s.t. \ & \mathcal{R}_k(\delta_i, G)\delta_{i,k} \geq 0 \quad \forall k, \\ & \delta_{i,k} \in \{0,1\} \quad \forall k. \end{aligned} \tag{3.29}$$

In this chapter, we show that every term of the optimization problem (3.29) can be expressed as smooth functions that depend on δ_i values. Hence, we can write the more comprehensive problem as:

$$\begin{aligned} \max_{\delta_i} \ & \pi_i\Big(N_i\big(\mathcal{G}_i(\delta_i, G)\big), D_i\big(\mathcal{G}_i(\delta_i, G)\big)\Big), \\ s.t. \ & \Bigg(\pi_k\Big(N_k\big(\mathcal{G}_i(\delta_i, G)\big), D_k\big(\mathcal{G}_i(\delta_i, G)\big)\Big) \\ & - \pi_k\Big(N_k\big(\mathcal{G}_i(0, G)\big), D_k\big(\mathcal{G}_i(0, G)\big)\Big)\Bigg)\delta_{i,k} \geq 0 \quad \forall k, \\ & \sum_k \operatorname{sgn}\big((\mathcal{G}_i(0,G) + \mathbf{I})^{J-1}\big)_{j,k}\delta_{i,k} \leq 1 \quad \forall j, \\ & \delta_{i,k} \in \{0,1\} \quad \forall k, \end{aligned} \tag{3.30}$$

or simply:

$$\begin{aligned} \max_{\delta_i} \ & \pi_i(\delta_i, G), \\ s.t. \ & \mathcal{R}_k(\delta_i, G)\delta_{i,k} \geq 0 \quad \forall k, \\ & B(\delta_i, G)_j \leq 1 \quad \forall j, \\ & \delta_{i,k} \in \{0,1\} \quad \forall k. \end{aligned} \tag{3.31}$$

By using our interpolation method, we are able to obtain an optimization problem based on the problem given in Section 2.7 that yields meaningful intermediate values for relaxed δ_i. This allows us to use deterministic non-linear mixed integer optimization techniques to solve this problem. During these computations, the functions for this parameter are assembled using the Matlab code in Appendix A.2. We note that for a constant G, the value of $B(\delta_i, G)_j$ depends linearly on δ_i, which is the case when we are solving (3.31).

3.3 Analysis of the Continuous Objective Function

In Section 3.2, we state that in order to solve our mixed integer problem with a non-linear objective function, we need to relax the integer constraints. Another important thing we need to consider is whether the objective function is concave when maximising the relaxed variables. In this section, we will analyse the profit function's second-order properties that depend on $\delta_i \in [0,1]^{J-1}$ by finding its Hessian matrix.

Throughout this section, $x \in \mathbb{R}^{J-1}$ is substituted for the continuous δ_i for simplicity. We use $n(x) \in \mathbb{R}$ and $m(x) \in \mathbb{R}$ respectively for the continuous equivalents of N_i and D_i that depend on the continuous decision variable [2].

3.3.1 Second Order Analysis

First we can write the continuous profit function by substituting $n(x)$ and $m(x)$ to (2.70):

$$\pi_i(n(x), m(x)) = \frac{(n(x)P - \beta_i)^2}{4n(x)^2\gamma_i} - d_i(n(x) - 1) - c_i m(x). \tag{3.32}$$

Along with our expression for $n(x)$ (given in (3.6)), finding the second derivatives of this expression may get complex. To avoid this, we can use the chain rule:

$$\frac{\partial^2 \pi_i}{\partial x_a \partial x_b} = \sum_k \left(\frac{\partial \pi_i}{\partial u_g} \frac{\partial^2 u_g}{\partial x_a \partial x_b} \right) + \sum_{g,h} \left(\frac{\partial^2 \pi_i}{\partial u_g \partial u_h} \frac{\partial u_g}{\partial x_a} \frac{\partial u_h}{\partial x_b} \right). \tag{3.33}$$

Here we define three new indices, u_g and u_h are used to represent $u_1 = n$ and $u_2 = m$. While x_a and x_b are two indices that represent k in $\delta_{i,k}$. Having established the notation, we can move on to finding the appropriate values.

First, we take n and m as independent variables and find the corresponding gradient and the Hessian:

$$\nabla \pi_i(n, m) = \begin{bmatrix} \frac{\partial \pi_i}{\partial n} \\ \frac{\partial \pi_i}{\partial m} \end{bmatrix} = \begin{bmatrix} -\frac{\beta_i{}^2 - P\beta_i n + 2d_i\gamma_i n^3}{2\gamma_i n^3} \\ -c_i \end{bmatrix}, \tag{3.34}$$

$$H = \begin{bmatrix} \frac{\partial^2 \pi_i}{\partial n^2} & \frac{\partial^2 \pi_i}{\partial n \partial m} \\ \frac{\partial^2 \pi_i}{\partial m \partial n} & \frac{\partial^2 \pi_i}{\partial m^2} \end{bmatrix} = \begin{bmatrix} \frac{\beta_i(3\beta_i - 2Pn)}{2\gamma_i n^4} & 0 \\ 0 & 0 \end{bmatrix}. \tag{3.35}$$

[2] We use the component size and direct connection values for the firm that is making the connection.

Next, we find the expression for $\frac{\partial n}{\partial x_a}$. To do this, we can write the expression for $n(x)$ as:

$$n(x)_i = N_i^0 + \sum_a M_{i,a} x_a. \tag{3.36}$$

The Jacobian of n can be found as the rows of M. For simplicity, let us define v as the transpose of the i^{th} row of M. That is to say: $v_a = M_{i,a}$.

Therefore we can write the first derivative of n as $\frac{\partial n}{\partial x_a} = v_a$. Moreover, for m, the continuous equivalent of direct connections, we have $\frac{\partial m}{\partial x_a} = 1$.

We see that the second derivatives of n and m are zeros as both the values of v_a and 1 are constant. It follows that:

$$\frac{\partial^2 u_g}{\partial x_a \partial x_b} = 0, \tag{3.37}$$

and we know that only $\frac{\partial^2 \pi_i}{\partial n^2} \neq 0$. So we work with a simpler version of the chain rule:

$$\frac{\partial^2 \pi_i}{\partial x_a \partial x_b} = \frac{\partial^2 \pi_i}{\partial n^2} \frac{\partial n}{\partial x_a} \frac{\partial n}{\partial x_b}. \tag{3.38}$$

Hence we can write the Hessian of continuous variables (H) as follows:

$$H_{a,b} = \frac{\beta_i \left(3\beta_i - 2Pn(x)\right)}{2\gamma_i n(x)^4} v_a v_b. \tag{3.39}$$

The first thing to note is that v_a and v_b are always non-negative as the values in M are all differences in making a single connection. Which by definition cannot decrease the size of a component. Moreover, for the case of i in particular, since it starts with no connections, making any single connection increases its component size. This shows that all values of v are positive. In the closed matrix form, we can write (3.39) as:

$$H = \frac{\beta_i \left(3\beta_i - 2Pn(x)\right)}{2\gamma_i n(x)^4} v v^T. \tag{3.40}$$

The expression:

$$\frac{\beta_i \left(3\beta_i - 2Pn(x)\right)}{2\gamma_i n(x)^4}, \tag{3.41}$$

is a coefficient of the matrix and since $n(x) \geq 1$ and $P > 1.5\beta_i$ it is always negative. Therefore, if we can show that $-vv^T$ is negative semi-definite, we can say that the profit function that depends on x is concave. To show this, we can consider the definition of negative semi-definiteness. A matrix A is negative semi-definite if $x^T A x \leq 0$ for all x. So let us substitute A with $-vv^T$:

$$x^T(-vv^T)x = -x^T v v^T x = -(v^T x)^T (v^T x). \tag{3.42}$$

We see that $(v^T x)^T (v^T x)$ is always no less than zero for any $v^T x$ with real values. Hence $x^T (vv^T) x$ is always non-negative, and $-x^T (vv^T) x$ is therefore always non-positive.

We have thus shown that $\pi_i(x)$ is a concave function for the interval $x \in [0, 1]^{J-1}$. This solution, of course, needs not be unique. Integer constraints severely reduce the number of solutions, but even then, multiple integer value sets of x may produce the maximum value.

3.3.2 First Order Analysis

Next, we apply the first-order condition to understand the solution's nature better. We recall the chain rule:

$$\frac{\partial \pi_i}{\partial x_a} = \sum_k \frac{\partial \pi_i}{\partial u_k} \frac{\partial u_k}{\partial x_a}. \tag{3.43}$$

We know that $\frac{\partial n}{\partial x_a} = v_a$ and that $\frac{\partial m}{\partial x} = 1$. Then we have:

$$\frac{\partial \pi_i}{\partial x_a} = -v_a \frac{{\beta_i}^2 - P\beta_i n(x)}{2\gamma_i n(x)^3} - v_a d_i - c_i. \tag{3.44}$$

For a continuous x, if $\hat{x}$ that equates (3.44) exists in the interval $x \in [0, 1]^{J-1}$ that $\hat{x}$ is a maximiser. And if not, the solution would lie on the borders of the feasible region.

In this section, we have shown the concavity of the objective function. In the next section, we discuss how we make use of the relaxed variables to find the solution to our problem.

3.4 Modifications to the Branch and Bound Algorithm

In Sections 3.2 and 3.3, we discuss how we can interpolate the intermediate values of the objective function for the relaxed decision variables and show that this function is concave within the relaxation interval $[0, 1]$. In this section, we use the findings of previous sections to solve our optimization problem.

3.4.1 Simple Branch and Bound Algorithm

The problems that can be classified as mixed integer non-linear programming may arise from different fields with properties unique to the situation they model. Our problem given in (3.31) is concave for relaxed variables and has a non-linear constraint that we call the consent condition. In order to find deterministic solutions for concave maximization problems (or convex minimization problems) with non-linear objective functions, tree-based methods are considered appropriate [7]. There are commercially available solvers for MINLP's, and approximating our profit to a smooth function, we can employ one to solve our problem. However, often for the mixed integer problems, building a solver that fits the exact problem yields solutions more efficiently. A general solver may still be preferable when the model is solved for specific values, but for general testing we consider the ability to tweak the solver an integral requirement for efficient computation.

Matlab offers a deterministic solver for linear mixed integer problems. However, for non-linear cases, Matlab only offers stochastic methods. Unlike Matlab however, python does offer solvers for MINLP problems, such as the library Gekko [6]. However, due to the matrix heavy calculations involved in our solution, and the importance of solver speed since it to be used for a large number of times during a multiple connectivity game, we consider it simpler to write the a branch and bound solver for the optimization problems in this thesis.

The branch and bound algorithm is a fairly robust method for solving mixed integer problems. Its principle idea is solving the equivalent problem obtained by converting integer variables to continuous variables that are constrained to the interval bound by integers (such as $x \in \{0, 1\}$ to $x \in [0, 1]$). This process is called variable relaxation, and the problem obtained this way is called the relaxed problem. Relaxed problem is solved using a method suitable to the problem. If the solution happens to also meet the integer criterion, the search is terminated. If not, one of the integer variables is taken, and sub-problems where each possible integer value of that variable is taken as constant, and the process is repeated for sub-problems [16]. This idea is also given in Algorithm 1 for a problem to find maximizer for $\pi_i(x)$ where $x \in \{1, 0\}^{J-1}$.

This particular implementation of the branch and bound algorithm is recursive and depth-first. This results in the algorithm checking every possible x unless an integer solution is reached earlier. Our sub-problems are non-linear, hence finding a solution to our sub-problems is a non trivial-task. Therefore, we are motivated to reduce the number of times we need to solve a sub-problem. This is achievable by using a slightly more complex version of the algorithm. This algorithm involves implementing active and inactive lists along with pruning. We explain these concepts in Sections 3.4.2 and 3.4.3.

Algorithm 1 Simple branch and bound algorithm for maximizing $\pi_i(x)$ for $x \in \{0,1\}^{J-1}$ [16].

```
Find $\hat{x} \in [0,1]^{J-1}$ that maximizes $\pi_i(x)$ with a suitable method.
if $\hat{x} \in \{0,1\}^{J-1}$ then
    return $\hat{x}$ .
else
    Create the sub-problem with $\pi_i^{(0)}(x_{2,\dots}) = \pi_i(0, x_{2,\dots})$.
    Solve the sub-problem with the Simple B&B Algorithm to find $\hat{x}^{(0)}$.
    Create the sub-problem with $\pi_i^{(1)}(x_{2,\dots}) = \pi_i(1, x_{2,\dots})$.
    Solve the sub-problem with the Simple B&B Algorithm to find $\hat{x}^{(1)}$.
    if $\pi_i^{(0)}(\hat{x}^{(0)}) > \pi_i^{(1)}(\hat{x}^{(1)})$ then
        return $\hat{x}^{(0)}$.
    else
        return $\hat{x}^{(1)}$.
    end if
end if
```

3.4.2 Branch and Bound with Active and Inactive Lists

The idea for active and inactive list implementation is that there are two lists, one for active and one for inactive solutions. If a solution meets the integer criteria, it is recorded in the inactive list. If not, it is added to the active list. At every iteration, one item from the active list is split into two sub-problems. Finally, the sub-problems are added to the relevant list depending on whether they meet the integer criteria or not.

Sub-problem generation is handled via static and dynamic parts of the solution. The static part is implemented from the first index, and only the dynamic part of x is maximized. A sub-problem is generated by taking the static part and adding 0 and 1 to its ends. When the active list is exhausted after no more than 2^{J-1} iterations (the maximum possible number of sub-problems), the x, which yields the highest value from the objective function, is the solution to the mixed integer problem. This is given in Algorithm 2. In order to start the inactive list, we need a candidate solution that meets the integer criterion and is feasible. In Section 2.6 we state that a vector of zeros is such a candidate, hence we use it for the algorithm.

This algorithm is less elegant compared to Algorithm 1, but with the introduction of pruning, it allows us to compute the result more efficiently.

Algorithm 2 Branch and bound algorithm with active and inactive lists [14].

```
Define active list $l_a$ and start it with the fully relaxed solution.
Define inactive list $l_d$ and start with $x = 0$.
for $2^K$ turns do
    Take $x_e$ from the end of $l_a$.
    Split $x_e$ to its static and dynamic part $s_e$, $s_a$.
    Define vectors: $s_0 = \begin{bmatrix} s_e \\ 0 \end{bmatrix}$, $s_1 = \begin{bmatrix} s_e \\ 1 \end{bmatrix}$.
    for $s = s_0, s_1$ do:
        Solve $\pi_i(s, s_a)$ where $s_a$ is the remaining dynamic part.
        if $\hat{x} = [s, s_a]$ meets the integer criteria then
            Add $\hat{x}$ to inactive list.
        else
            Add $\hat{x}$ to active list.
        end if
    end for
    if Active list is empty then
        Break.
    end if
end for
The solution is $\hat{x}$ with the largest $\pi_i(\hat{x})$ value from the inactive list.
```

3.4.3 Pruning

While we are solving for an integer x using a branch and bound algorithm with the approach given in Algorithm 2, we would end up with a large number of items in the active and inactive lists. As the values in the active list are maximisers for the entire boundary [0,1], then their $\pi_i(x)$ must be no less than the maximum of the objective function π_i for $x \in \{0,1\}$. Then for x_a as an item from the active list and x_d as an item from the inactive list, if $\pi_i(x_a) < \pi_i(x_d)$, so must every other result from sub-problems derived from that x_a. It follows that we can prematurely terminate the branch of x_a by removing it from the active list. This process allows us to reduce the computation time, and it is called pruning [16].

Since we already know that the profit from $\hat{x}$ calculated here is later compared with the profit from the original adjacency matrix G_{n-1} before firm i proposing the alteration, we can calculate that value early on and use it for additional pruning of active branches.

3.4.4 Consent and Component Conditions

Up to this section, we have acted like the integer constraint is the only constraint in our problem (3.31). However, we have the consent and component conditions:

$$\mathcal{R}_k(\delta_i, G)\delta_{i,k} \geq 0 \quad \forall k, \tag{3.45a}$$

$$B(\delta_i, G)_k \leq 1 \quad \forall k. \tag{3.45b}$$

There are two possible approaches we can take for these conditions. First, we can add them as constraints to the relaxed sub-problem obtained in every iteration of the branch and bound algorithm. Alternatively, we can solve sub-problems without these constraints and check the solutions in a different step within our algorithm. In this section, we discuss how we can apply the latter approach. In Section 4.2.2, we discuss the advantages of the latter approach over the former with computation times.

Our first inclination is using consent and component conditions for filtering the solutions in the inactive list. However, when we do not include these conditions in the relaxed problem, it may be possible for a solution to not meet them, while some other solution at one of the sub-problems that would be generated from that solution does meet them and yield a lower profit. Hence we cannot simply filter the inactive list for these conditions.

Instead, we consider the constraints at the step of the sub-problem evaluation, where we decide if integer criteria are met. If the condition is met and constraints are fulfilled, we may add $\hat{x}$ to the inactive list. However, if it is not, and if the active part is not empty (static part does not include all the δ_i values), we add $\hat{x}$ back to the active list, while if it is empty, $\hat{x}$ is removed.

However, there is a second thing we can do for the component condition in particular. In a sub-problem, we only evaluate the active part, but the static part may not fulfil the component condition even when all the active part is zero. In such a case, what we did above keeps sending that $\hat{x}$ to the active list while it can never be feasible. So, before solving the sub-problem, we can check whether $\pi_i(s, 0)$ (s is the static part for that sub-problem) is feasible according to the component condition. If not, we can simply skip that sub-problem.

Before finalizing the algorithm, we need to consider a few more points. First is how we will implement the integer condition. Since we are using computers, the results we obtain may not be exactly 1 or 0, but very close. We use an integer tolerance of $\epsilon = 10^{-8}$ for the computations in this thesis to ensure such values are still considered integers. Hence, we take $\delta_{i,k} \leq \epsilon$, as 0; $\delta_{i,k} \geq 1 - \epsilon$ as 1, and any values in-between do not meet the integer constraint.

In the numerical simulations, we need to consider how to pick an item from the active list. One approach could be always picking the item that yields the highest $\pi_i(x)$, thereby making the search more breadth oriented. Nevertheless, during testing, we find that this implementation finds useful values for pruning in a much later stage of the computation, and hence we find the depth-first approach preferable when looking for pruning values.

Additionally, we need to consider how to break the tie if multiple solutions have the same profit. In real life, such choices may be made with external considerations, but in this study, we simply pick the last item in the sorted list.

3.4.5 Modified Algorithm

Algorithm 3 Modified branch and bound.

```
Start with $p_b$ as the baseline profit.
Define active list $l_a$ and start it with the fully relaxed solution.
Define inactive list $l_d$ and start with $x = 0$.
for $2^K$ iterations do
    Find the largest $\pi_i(x_d)$ from $l_d$.
    Remove $x_a$ from $l_a$ if $\pi_i(x_a) < \max(\pi_i(x_d), p_b)$.
    Take $x_e$ from the end of $l_a$ and split $x_e$ to its static and active part $s_e$, $s_a$.
    Define vectors: $s_0 = \begin{bmatrix} s_e \\ 0 \end{bmatrix}$, $s_1 = \begin{bmatrix} s_e \\ 1 \end{bmatrix}$.
    for $s = s_0, s_1$ do
        $z = [s, 0, 0, \dots]$.
        if $\sum_{j \neq k} z_k sgn\big((\mathcal{G}(z, G_n) + \mathbf{I})^{J-1}\big)_{k,j} > 1$ for any $k$ then
            Continue.
        end if
        Solve $\pi(s, s_a)$ where $s_a$ is the remaining active part.
        if $\hat{x} = [s, s_a]$ meets the integer criteria then
            $\mathcal{R}_k(\delta_i, G)\delta_{i,k} = \delta_{i,k}\Big(\pi_k(\hat{x}) - \pi_k(0)\Big)$ .
            $B(\delta_i, G)_j = \sum_k sgn\big((\mathcal{G}(0, G_n) + \mathbf{I})^{J-1}\big)_{j,k}\delta_{i,k}$.
            if $\mathcal{R}_k(\delta_i, G)\delta_{i,k} \geq 0$ for all $k$ and $B(\delta_i, G)_j \leq 1$ for all $j$ then
                Add $\hat{x}$ to inactive list.
            else if $s_a$ is not empty then
                Add to the active list.
            end if
        else
            Add $\hat{x}$ to active list.
        end if
    end for
    If the active list is empty, break.
end for
The solution is the largest $\pi_i(x)$ from the inactive list.
```

With all the discussed considerations, we are left with our Modified Branch and Bound scheme given in Algorithm 3, which we've purpose-built to solve our problem. Our application of this algorithm in Matlab can be found in Appendix A.5.

The final remaining question regarding our branch and bound scheme is how we solve the relaxed sub-problems created by the algorithm. Since we have a non-linear objective function, we can use Matlab's fmincon function. How to pick the parameters for fmincon is discussed in detail in Section 4.2.1.

3.5 Procedure to Find Stable Systems in MCG

In the final section of this chapter, we present a procedure that finds systems that are stable when MCG is played. This procedure is partly inspired by the work of Watts [44] where a dynamic game of network formation is discussed. In that game, possible connections are evaluated in turns, and if a connection is profitable to both parties, the network is updated to include it. Finally, when no possible pairwise alteration is found, the game is terminated. Similarly, in our procedure, we start from an unconnected system ($G_{i,j} = 0$ for all i, j) and assess if any firm profits from altering the network with an MC move. If a profit-increasing alteration is found, this process is repeated for the altered network. We represent this progression with the matrix series G_n. The series start with an unconnected adjacency matrix G_0, the network after the first alteration is G_1, the second alteration is G_2 and so on. This iterative process continues until a system is reached that no player can profit by altering according to the rules of MCG given in Section 2.7. The algorithm that performs this procedure is given in Algorithm 4, and it is applied with Matlab in Appendix A.8.

This procedure can terminate in three ways. The first way is it may not converge in a given number of cycles (C). Or indeed, at any number of cycles. Alternatively, it may reach MCG stability where no firm benefits from altering G_n. Thus, no firm alters the network structure for an entire cycle. Another possibility is the algorithm reaching cyclic progression, where firms do alter the network, but their choices cancel each other out, and the resultant system is the same at the end of the cycle.

Algorithm 4 Procedure to find Stable Systems in MCG.

```
Start with an empty matrix $G_0$ of $J$ rows and columns.
Define the number of cycles as $C$.
Given $P$, $\beta$, $\gamma$, $d$, $c$.
for $c = 1, 2 \ldots C$ do
    for $i = 1, 2, \ldots J$ do
        Find $N(\mathcal{G}(x, G_n))$ and $D(\mathcal{G}(x, G_n))$.
        Calculate baseline $p_b$ as $\pi(G_n)_i$.
        Assemble $\pi(x)$, $\mathcal{R}(x, G_n)$, and $B(x, G_n)$.
        Find the best $\hat{x}$ using MBB.
        if $\pi_b < \pi(\hat{x})$ then
            $G_{n+1} = \mathcal{G}(\hat{x}, G_n)$.
        else
            $G_{n+1} = G_n$.
        end if
        $n = n + 1$.
    end for
    if $G_n = G_{n-1} = \cdots = G_{n-J}$ then
        Return MCG stability.
    end if
    if $G_n = G_{n-J}$ then
        Return cyclic progression.
    end if
end for
```

We note that with this procedure, depending on the turn order we consider firms, different stable systems may emerge. This ordering can be fully random, with a permanent order, or with respect to a firm property such as considering the highest or lowest earning firm for a particular network. Moreover we cannot guarantee that with different turn orders all stable systems are reachable with this procedure. Therefore our claim is simply that with this procedure, we can find a network that is stable for a given set of firm parameters.

CHAPTER 4

SIMULATIONS OF THE MULTIPLE CONNECTIVITY MODEL

4.1 Introduction

In Chapter 3, we discuss how we can build the relaxable objective function $\pi_i(\delta_i)$ and how we can solve it. In this chapter, we run computer simulations to find stable systems for the Multiple Connectivity Game and discuss the various results we obtain. In Section 4.2 we check the effectiveness of our computational method and algorithm choices. After this, we continue our discussion of stable systems with MC and PW approaches in Section 4.3. Later, we do simulations in order to demonstrate how altering various firm parameters alters the model and the resultant stable networks. The simulations are done using the code given in Appendix A.17, and are sorted based on the number of firms involved as Sections 4.4 and 4.5.

4.2 Effectiveness of the Computation Methods

In this section, we solve the problem of finding optimal moves for a player using variations of our implementation to demonstrate its effectiveness. Appendix A.17 shows the codes used for this part.

4.2.1 Parameters for the Non-linear Solver

In order to solve the sub-problems we obtain via relaxing integer parameters, we use *fmincon* which uses an interior-point method as the default algorithm, but it is also possible to use sequential quadratic programming (SQP), and active set methods. In this section, we compare the computation time of these approaches with the modified branch and bound algorithm.

As a benchmark problem, we pick $P = 10$, $\beta = 5$, $\gamma = 2$, $d = 0.2$, $c = 1$ for all firms and look at the computation times of running the procedure to find stable systems for different values of J. The results obtained from the Matlab profiler are given in Figure 4.1. We see that the sequential quadratic programming algorithm solves subproblems faster than the alternatives in every given firm count. We run each simulation once, so the exact data points may vary. However, since algorithms are run many times during a single play, we can say that a wider trend is still applicable when considering which algorithm to use. For the rest of the simulations, we use "sqp" as the algorithm option for *fmincon*.

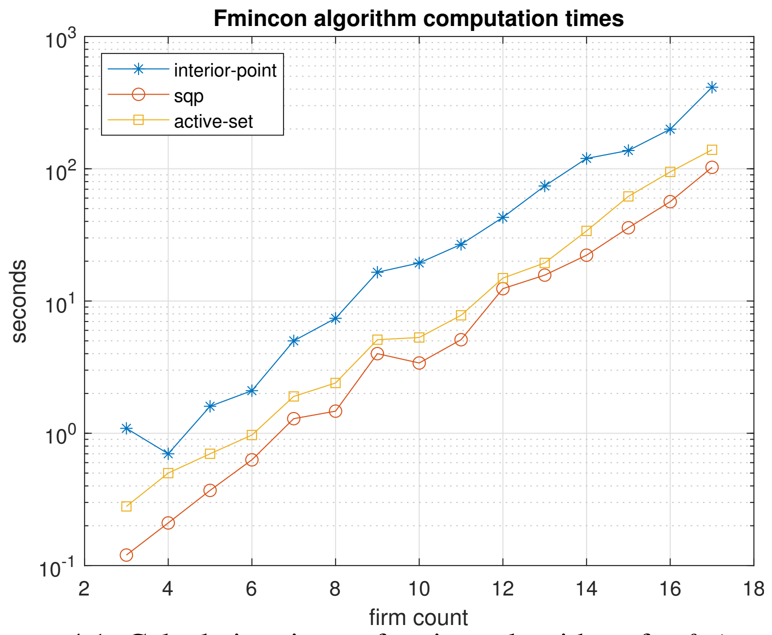


Figure 4.1: Calculation times of various algorithms for *fmincon*.

4.2.2 Modified and Pure Branch and Bound

The computation that takes the longest time in a B&B scheme is solving the sub-problems. However, in addition to picking the algorithm that solves the sub-problem fastest, the code performance can be improved by reducing the number of times a sub-problem needs to be solved. This can be done via means like pruning or premature termination by component condition, as discussed in Section 3.4.4. We achieve this by modifying the B&B scheme into the MBB. While both algorithms produce the same result, MBB aims to yield the result faster. Moreover, since we are solving a mixed integer problem where all variables are integer (pure integer problem), it is also possible for us to solve this problem by looking at each possible answer and comparing the obtained profit. This approach is called exhaustive searching, and we use it to solve the problem in Section 2.6. In this section, we compare these three approaches in terms of computation time with the benchmark parameters $P = 10$, $\beta = 5$, $\gamma = 2$, $d = 0.2$, and $c = 1$.

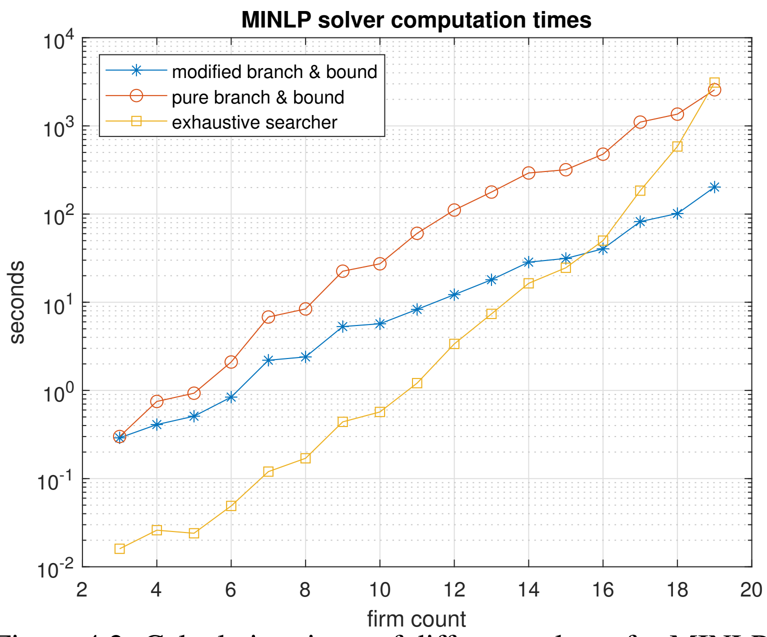


Figure 4.2: Calculation times of different solvers for MINLP.

In Figure 4.2, we can see that while exhaustive searcher works faster for a low number of firms, with higher firm numbers, it becomes prohibitively slow, reaching up to almost an hour for 19 firms. Meanwhile, our MBB algorithm works much faster for all firm counts compared to an implementation of the pure branch and bound algorithm. In all firm counts, three methods are called for the same number of times. Despite this, the pure branch and bound algorithm runs up to ten times slower for higher firm counts, further emphasising the necessity of modifying the algorithm for our problem.

4.3 Comparison of MC to PW Stability

In this section, we compare how MC and PW approaches lead to stable systems with component profit functions using simulations. We focus on systems where firm parameters are the same for all firms. In Section 2.5.6, we state that for firms with equal parameters, it is sufficient to check stability by only considering unique isomorphs. In Figure 4.3, unique isomorph forests with 5 nodes are given. To illustrate the difference between using the MC and PW approach for network formation, we apply various profit functions to these networks for the remainder of this section. The code used for this section is given in Appendix A.13.

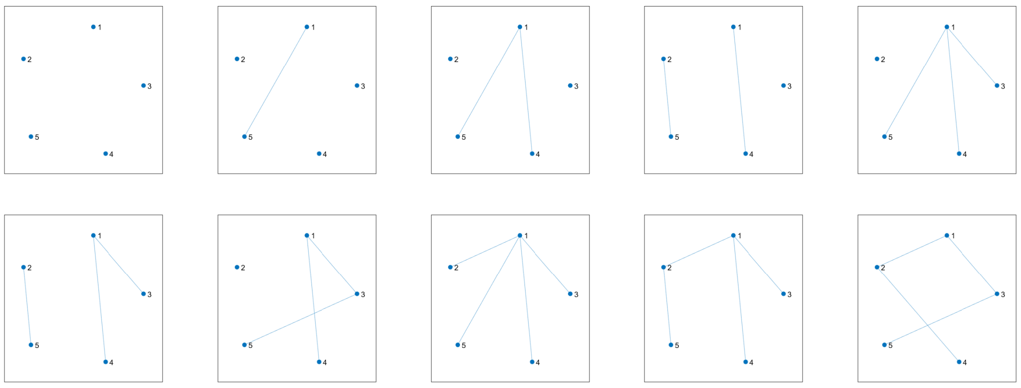


Figure 4.3: Unique isomorph forests with 5 nodes.

4.3.1 Large Stable Components

We start with $P = 10$, $\beta = 5$, $\gamma = 2$, $c = 1$, $d = 0.1$, the parameters used in the benchmark from the previous section problem. In Section 2.5, it is shown that trees larger than the optimal component size cannot be stable. The optimal component size for these parameters is 8, which suggests that stable networks will have components with no more nodes than 8. With 5 firms, this suggests that we can find stable single-component networks.

We compute stability for all unique isomorphs and display the stable networks in Figure 4.4 and unstable networks in Figure 4.5. In the figures, a node that can improve its profit is shown with triangle markers, and the profits firms make due to the network are given in parentheses. We see that all the networks with component counts of 4 and 5 are stable, while networks with smaller components are not. This shows us that it is beneficial for firms to merge into larger components with these parameters.

We can compare the MC results with PW results given in Figure 4.6. In the figure, we can see three types of connections. Black dashed lines are connections that do not exist in the network, but firms benefit by forming them. Thick red lines show existing connections that firms benefit by breaking. Finally, thin blue lines are connections that firms prefer maintaining. By looking at the results, it is clear that there are no pairwise stable systems with the given profit function. The MC stable networks are PW unstable because one party of each connection benefits from breaking it. The unilateral benefit also exists for the case with MC stability. However, the firms are not able to get the consent of their other connections when they alter a network by breaking a connection.

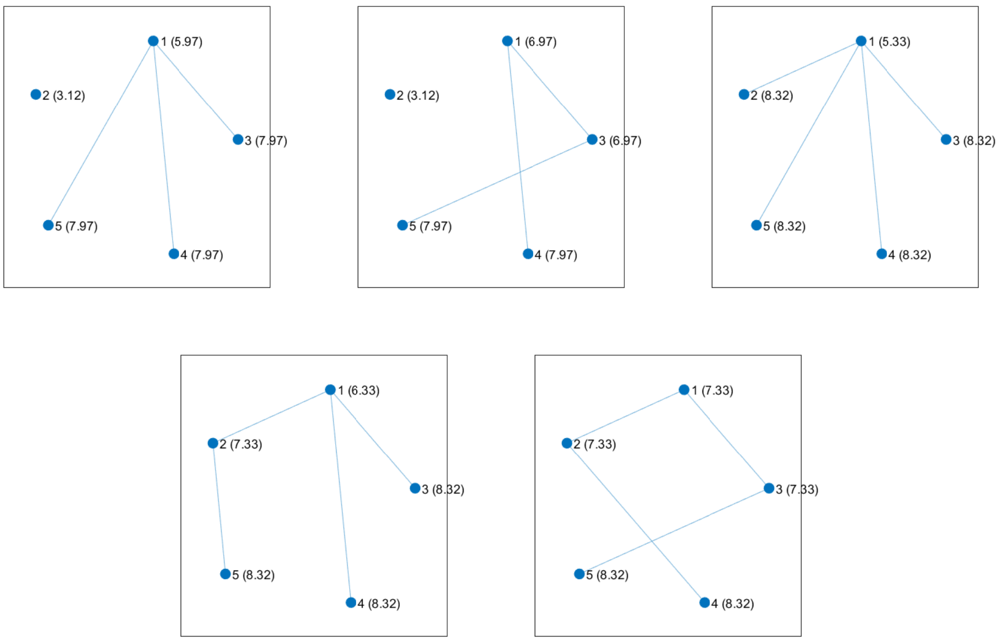


Figure 4.4: MC stable unique isomorph networks with 5 firms and equal parameters.

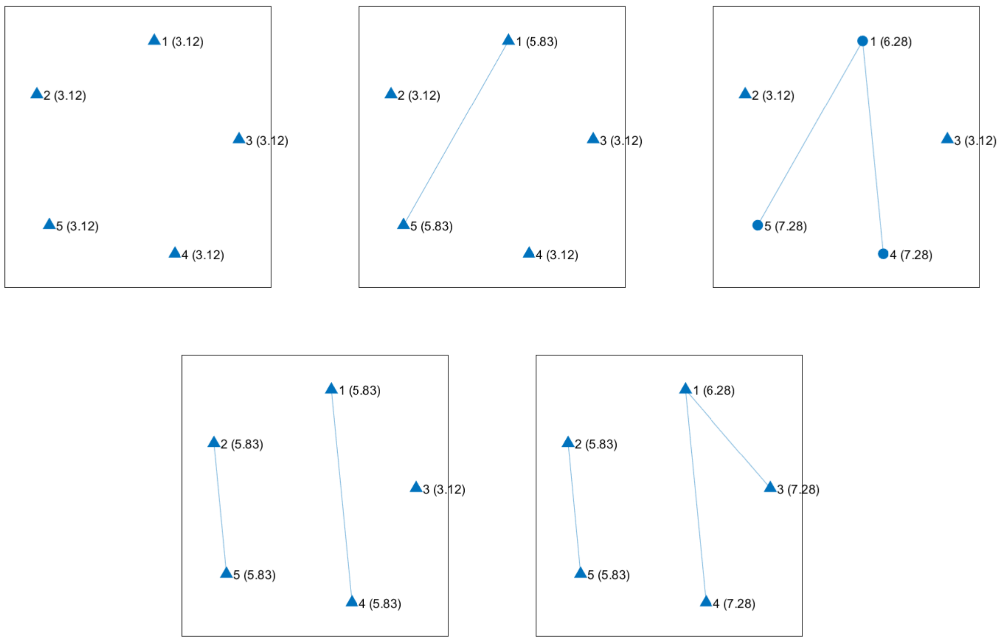


Figure 4.5: MC unstable unique isomorph networks with 5 firms and equal parameters.

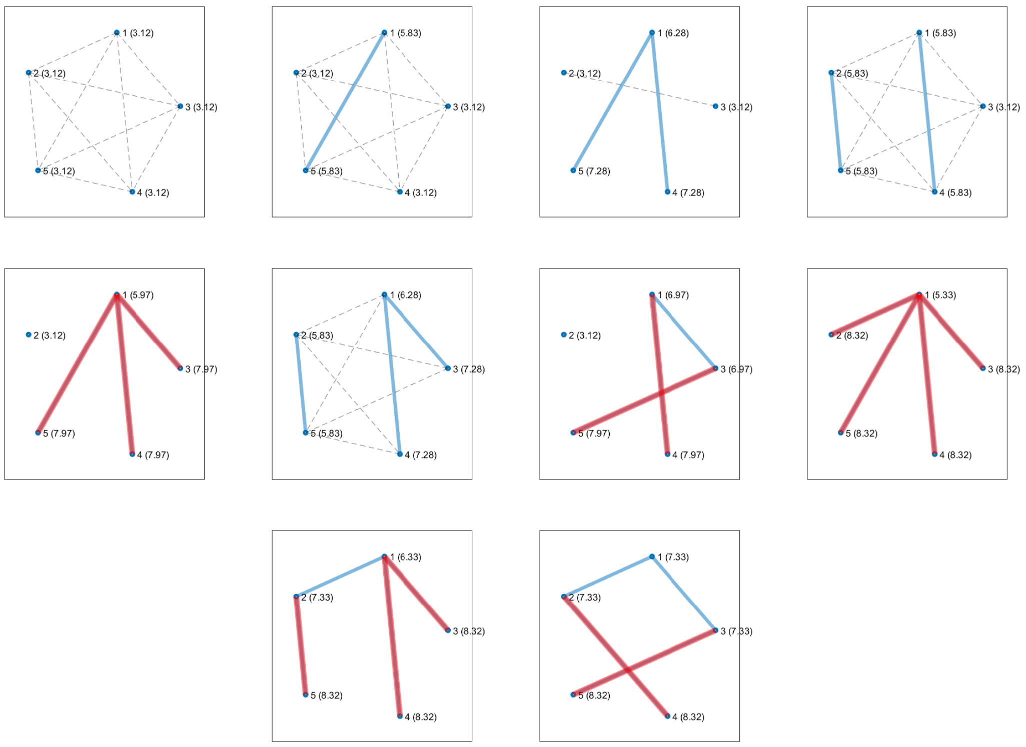


Figure 4.6: PW unstable unique isomorph networks with 5 firms and equal parameters.

4.3.2 Small Stable Components

In this section, we consider a smaller optimal component size of $\hat{N} = 3$, which can be done by altering firm parameters.

First, we can set $d = 1$ to set the optimal component size to $\hat{N} = 3$ while keeping others the same as done in Section 4.3.1. For these parameters, we have two MC stable networks, given in Figure 4.7 and one PW stable network, given in Figure 4.8. In addition, MC unstable networks are given in Figure 4.9 and PW unstable networks are given in Figure 4.10. Here, we see that having a lower optimal component size indeed results in larger networks being unstable. However, we can also observe that while a network of a three-firm chain and a pair is MC stable, it is PW unstable. This can be explained by considering the central node of the three-firm chain. In the centre, the firm's profit is 4.68, while a firm in a pair makes 5.03 profit. Clearly, $5.03 > 4.68$, so the centre firm benefits by severing a connection. However, the leaf node that stays connected to the centre suffers from this by losing 0.65 profit. Therefore, the leaf does not consent to the MC alteration, and the chain is also stable.

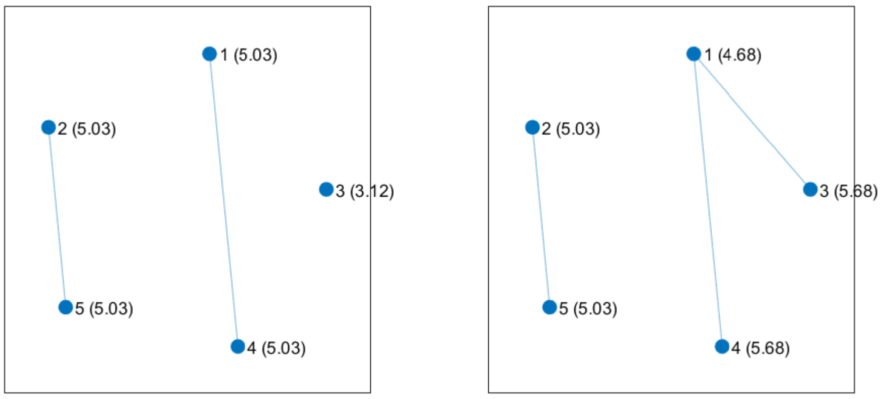


Figure 4.7: MC stable unique isomorph networks with 5 firms and $d = 1$.

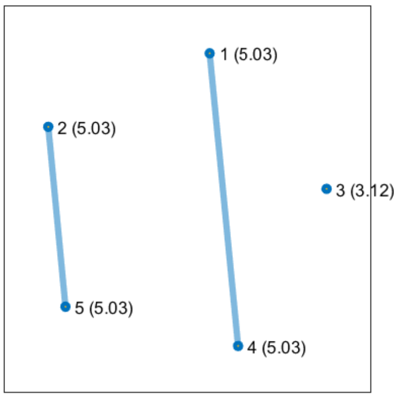


Figure 4.8: PW stable unique isomorph networks with 5 firms and $d = 1$.

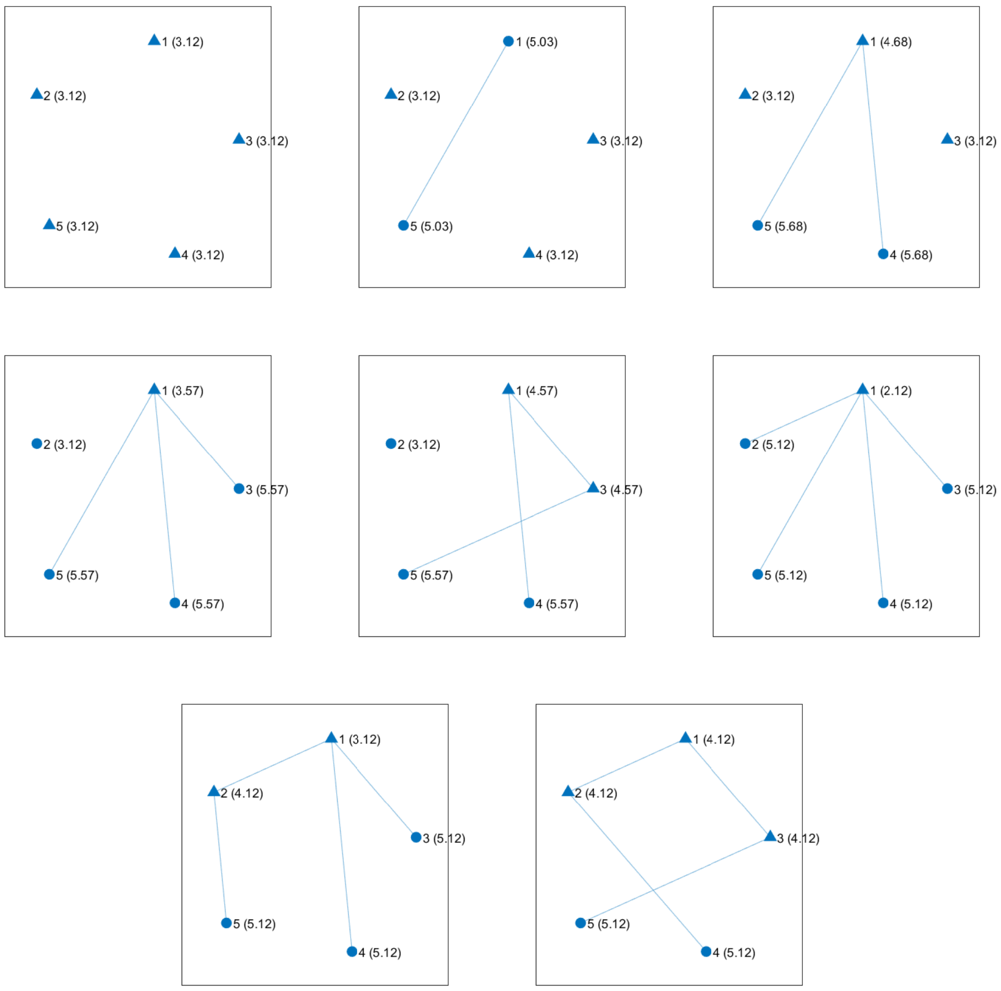


Figure 4.9: MC unstable unique isomorph networks with 5 firms and $d = 1$.

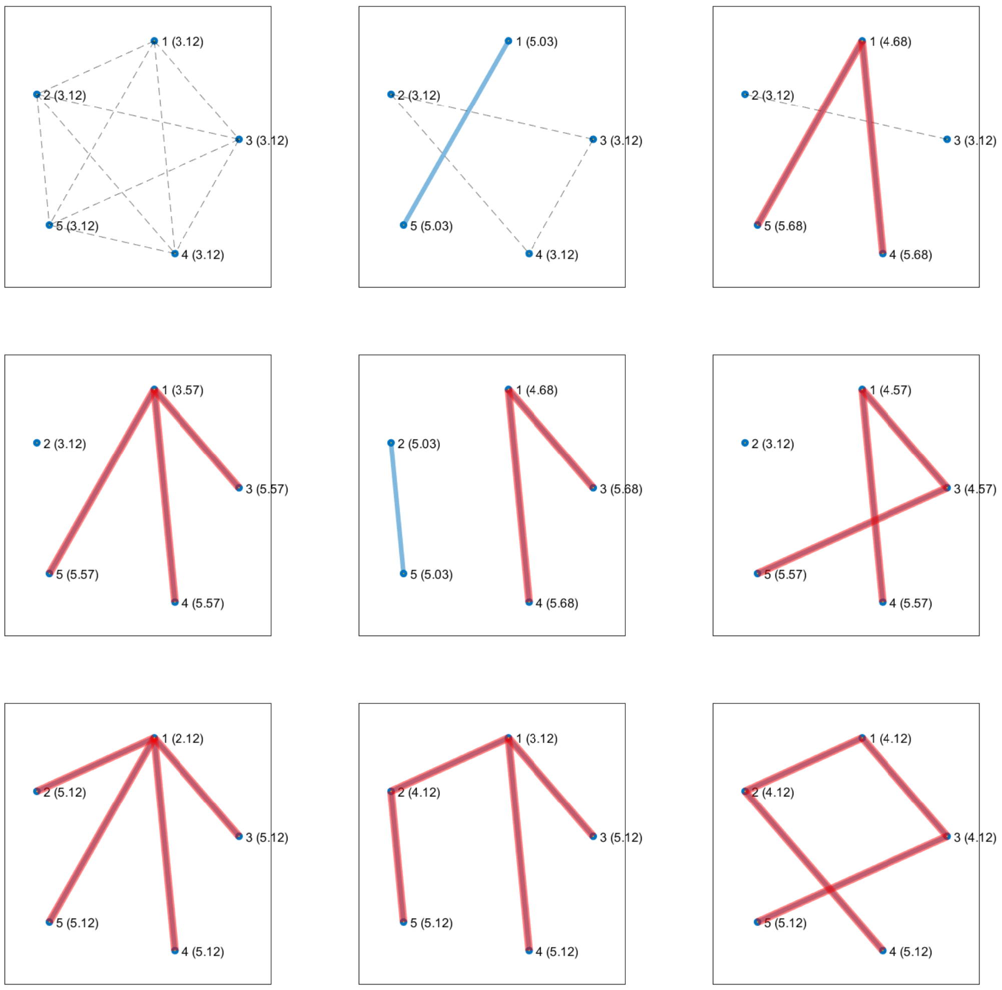


Figure 4.10: PW unstable unique isomorph networks with 5 firms and $d = 1$.

Another way to reach an optimal component size of $\hat{N} = 3$ is setting $\gamma = 20$. The MC stable system with these parameters is given in Figure 4.11, and the PW stable system is given in Figure 4.12. For these parameters, we see that PW and MC stable systems are identical, and they are unconnected systems. This result shows us that optimal component size is insufficient to predict stable systems, and the firm parameters should be investigated separately.

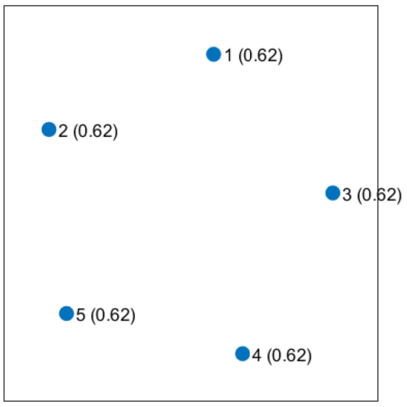


Figure 4.11: MC stable unique isomorph networks with 5 firms and $\gamma = 20$.

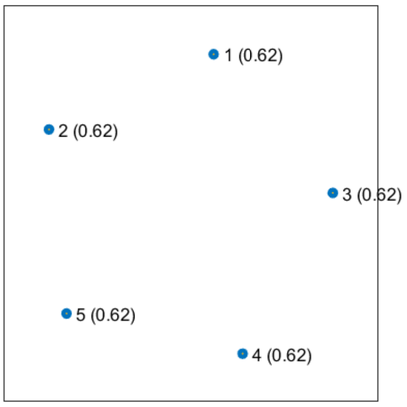


Figure 4.12: PW stable unique isomorph networks with 5 firms and $\gamma = 20$.

The examples in this section illustrate how MC and PW approaches result in different stable systems. Our first intuition may be that since MC allows for a larger set of moves, fewer systems would be MC stable. However, by constraining firms to only make alterations that require the consent of their direct connections, we are able to find a larger number of stable networks with the same profit. There are two ways to interpret this finding. First, firms that consider each connection separately may make choices that are ultimately against their best interest. The second is that considering each connection separately is unrealistic. We favour the latter interpretation since it is both possible and profitable for firms to consider multiple connections while deciding their interests. Therefore, we consider the multiple connectivity approach to be a more realistic prediction of short-term choice-makers in network formation. Moreover, we see that when stable systems are only the unconnected systems or systems of pairs, MC and PW approaches reach the same results.

4.4 Tests with Nine Firms

An analysis based on a small number of firms may not allow us to observe all the nuances of the MC approach for network formation. However, as the firm number increases, the number of unique isomorphic forests also increases. Therefore, instead of checking every unique isomorphic forest, we can use the algorithm we introduce at Section 3.5 and find an MC stable system for a given set of parameters. In this section, we do so for systems with 9 firms.

4.4.1 Equal Parameters

We start by considering $J = 9$ firms with the same parameters: $P = 10, \beta = 5, \gamma = 2$, $d = 0.2$, $c = 1$. The Matlab script for the 9 firm simulations with equal parameters is in Appendix A.18. The progression of the network can be traced in Figure 4.13 where the x-axis shows turns, the y-axis shows the firm number and darker squares indicate a higher profit for the firm. Moreover, in Figure 4.14, the unstable networks after each MC alteration can be observed. The resultant stable network has two stable components: a long chain and an unconnected node. The firms in the long chain, especially those on the edges, earn the largest profit from this configuration, while the unconnected generate less profit.

The moves that destabilise a 6 firm chain are not made by leaves attaching to new firms but by a centre firm replacing its shorter tail with a longer one outside. This phenomenon occurs at $n = 9$ when firm 9 breaks off 2 and 7 in favour of a 3-firm chain and again at $n = 11$ with firm 2. This is an interesting example of collective action generating higher payoff for individuals.

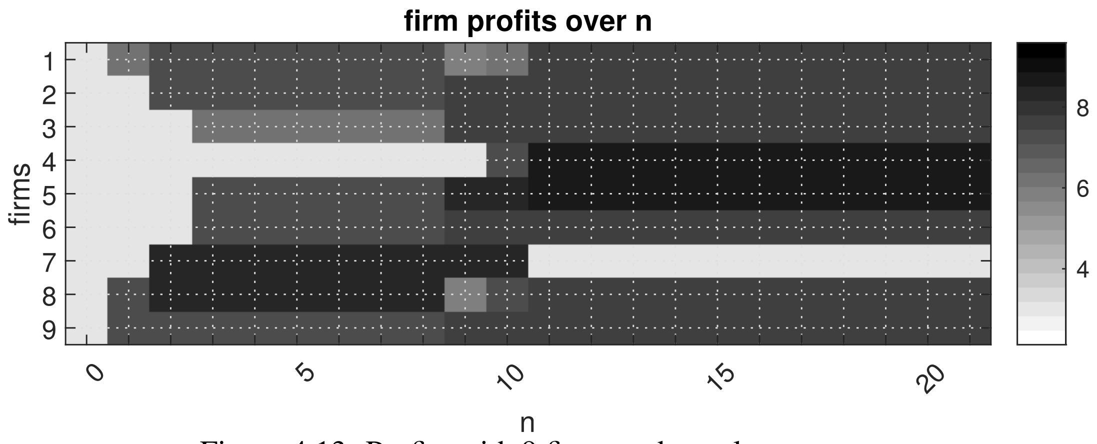


Figure 4.13: Profits with 9 firms and equal parameters.

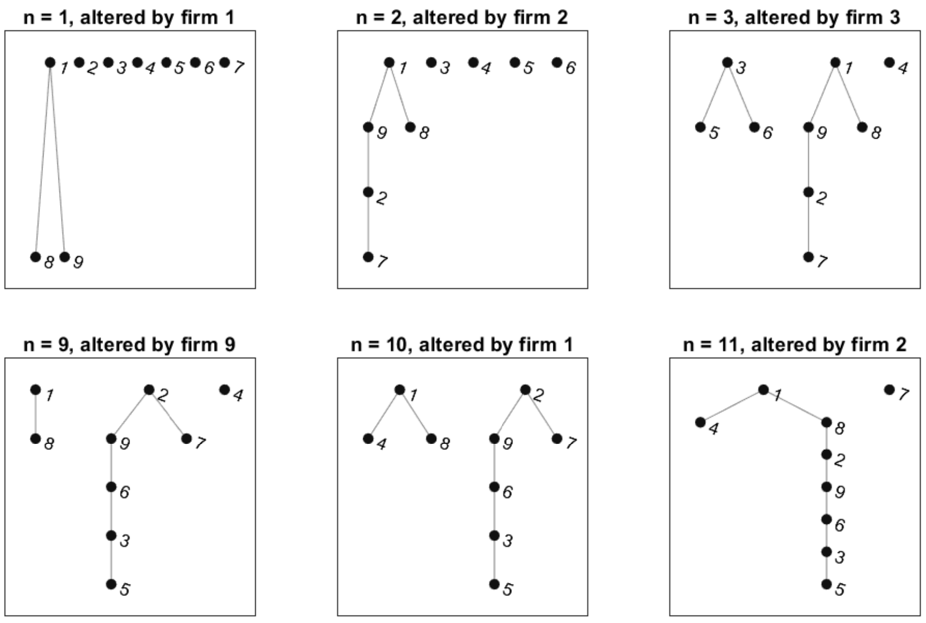


Figure 4.14: Network progression with 9 firms and equal parameters.

4.4.1.1 Shuffled Turn Order

Looking at the progression from Section 4.4, we can ask how significant turn order is on the resultant systems. To observe this, we can consider three other scenarios with the same parameters, except turn order is randomised at each cycle using different seeds. The stable networks obtained in this way are given in Figure 4.15. Interestingly, the stable networks contain a component of 8 firms, but unlike the stable network given in Figure 4.14 that component is not a chain.

Moreover, these networks all have a tree with 8 firms and a single unconnected firm. Since the optimal component size for these parameters is 8, this observation fits the ceiling condition we discuss in Section 2.5.5. However, the fact that outcomes are isomorphic is a mere coincidence.

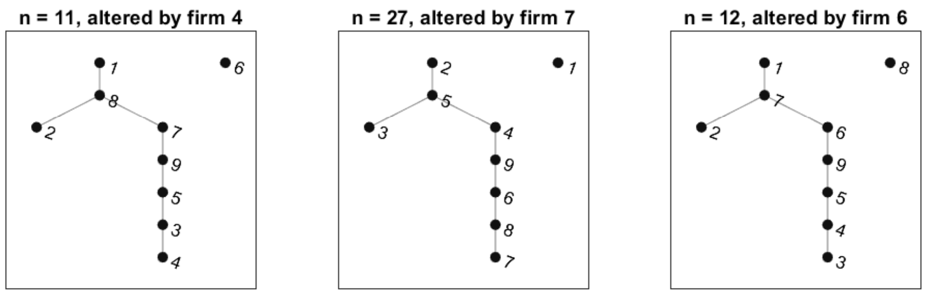

n = 11, altered by firm 4
1
6
8
2
7
9
5
3
4
n = 27, altered by firm 7
2
1
5
3
4
9
6
8
7
n = 12, altered by firm 6
1
8
7
2
6
9
5
4
3

Figure 4.15: Stable networks with 9 firms, equal parameters, and shuffled turn order with seeds 1, 2, 3 from left to right.

4.4.1.2 Alternate Linear Cost

Next, we consider the effect of altering model parameters, starting with the linear cost, as it is the term directly related to the network parameters. We first consider the effects of a larger $\beta = 6.66$ for all players, which is just below our concavity limit at $\beta < 1.5P$. The results of the simulations are shown in Figures 4.16 and 4.17. This set of parameters yields the optimal component size of 9. However, again, the players converge into a chain of 8 firms, and one firm is completely left out. With a higher β, we expect the cost difference of a larger component size to be more substantial; thus, it would be profitable to form a longer chain. However, even with this additional incentive, we see that no firm on the chain consents to connect with an unconnected node outside the chain.

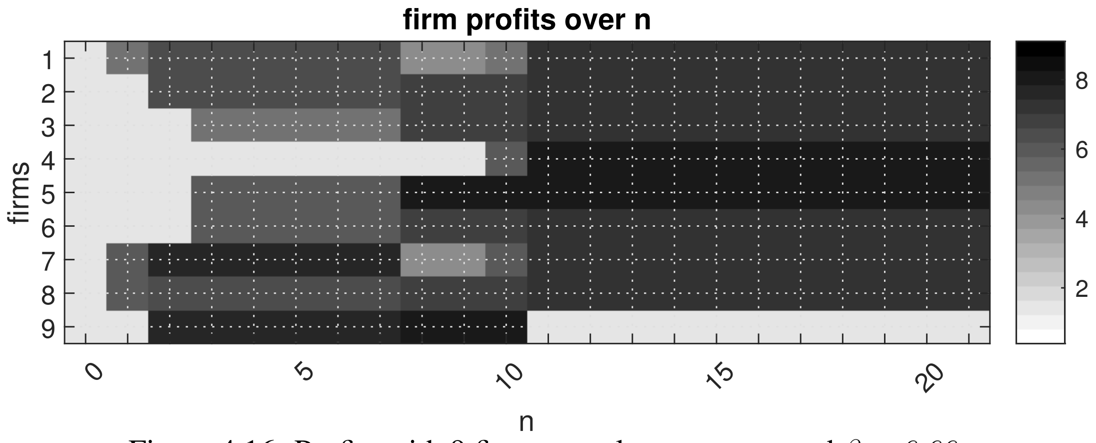


Figure 4.16: Profits with 9 firms, equal parameters, and $\beta = 6.66$.

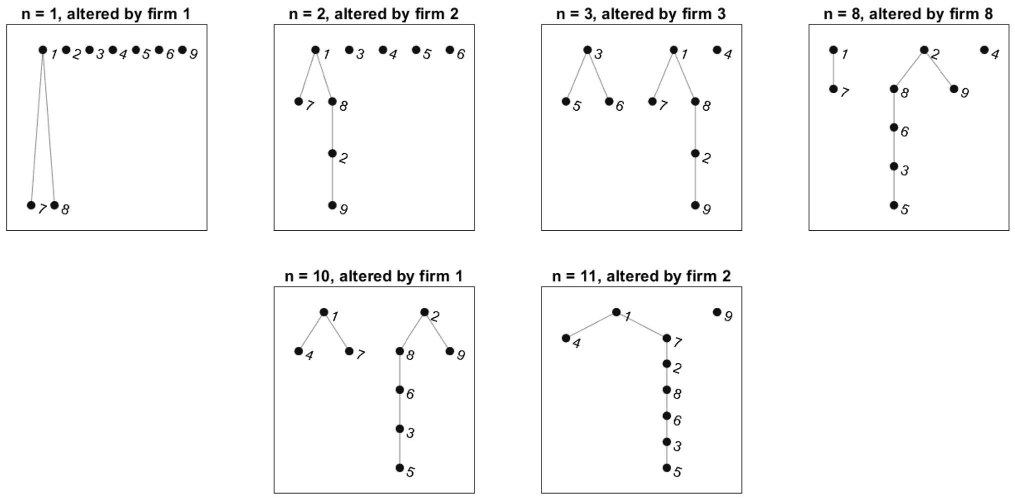


Figure 4.17: Network formation with 9 firms, equal parameters, and $\beta = 6.66$.

Conversely, when we look at the effects of a smaller $\beta = 3$ (and optimal component size of 6) for all players given in Figures 4.18 and 4.19, we see that the benefit from a cheaper production is surpassed by the cost from being in a larger component with less number of firms. Thus, we end up with two smaller chains and one unconnected firm.

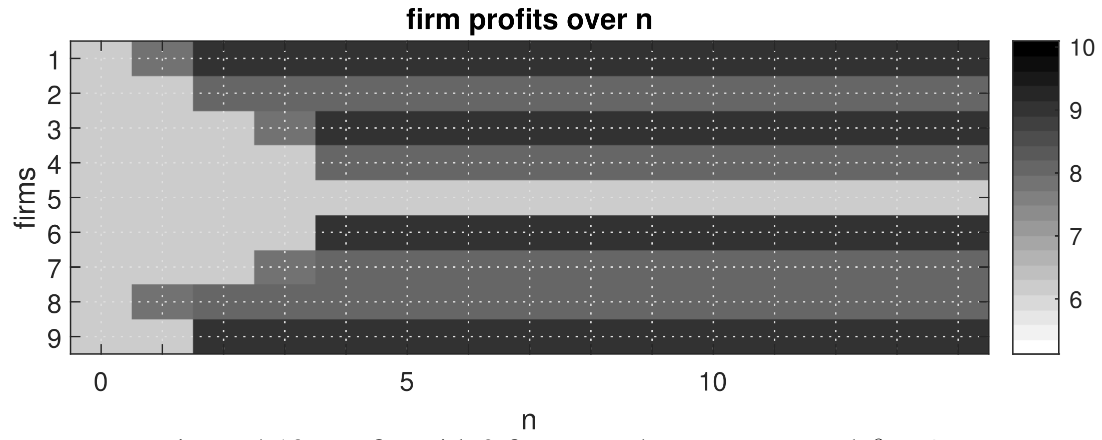


Figure 4.18: Profits with 9 firms equal parameters, and $\beta = 3$.

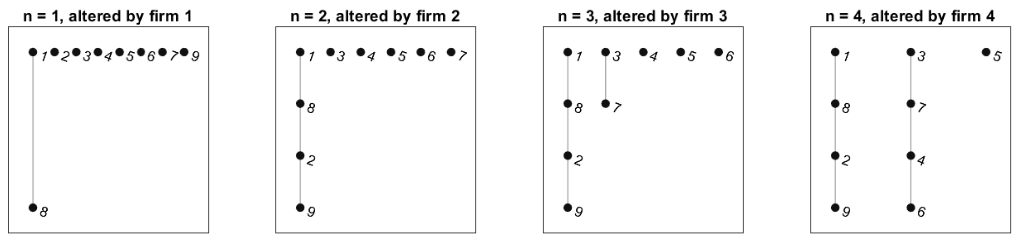


Figure 4.19: Network formation with 9 firms, equal parameters, and $\beta = 3$.

4.4.1.3 Alternate Quadratic Cost

We can also check how the quadratic cost influences network formation instead of the linear cost by setting $\beta = 5$ and altering γ. We start with $\gamma = 4$ for all firms given in Figures 4.20 and 4.21. Here, the higher quadratic cost, while not directly multiplied by component size, has a significant effect on the resultant components and no component larger than 2 firms can be formed under this condition.

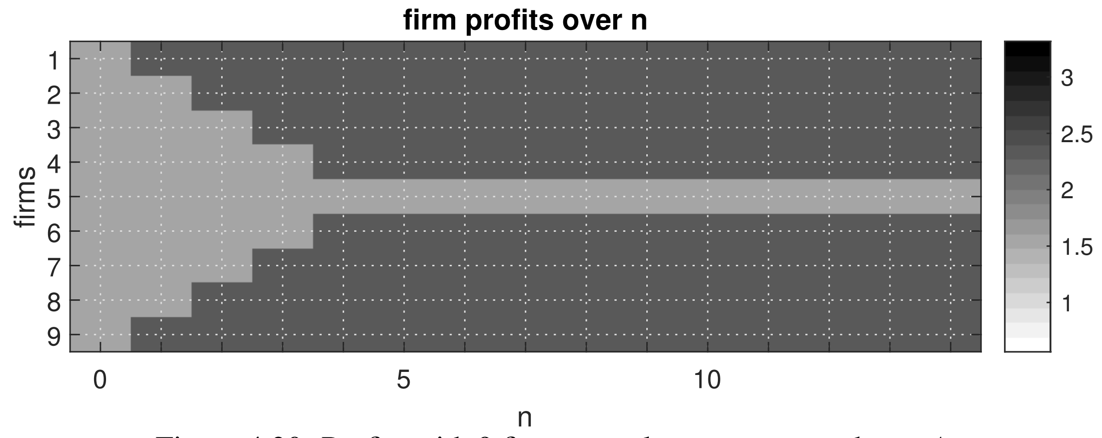


Figure 4.20: Profits with 9 firms, equal parameters, and $\gamma = 4$.

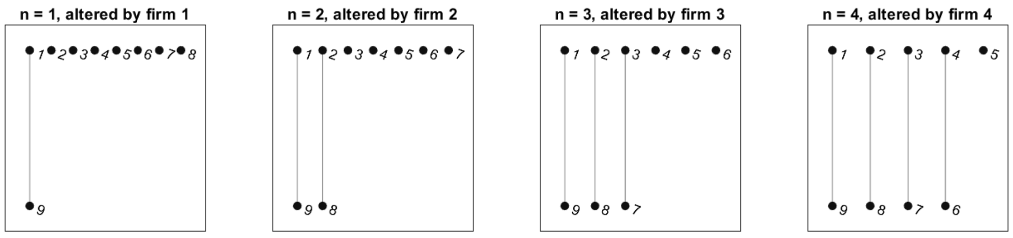


Figure 4.21: Network formation with 9 firms, equal parameters, and $\gamma = 4$.

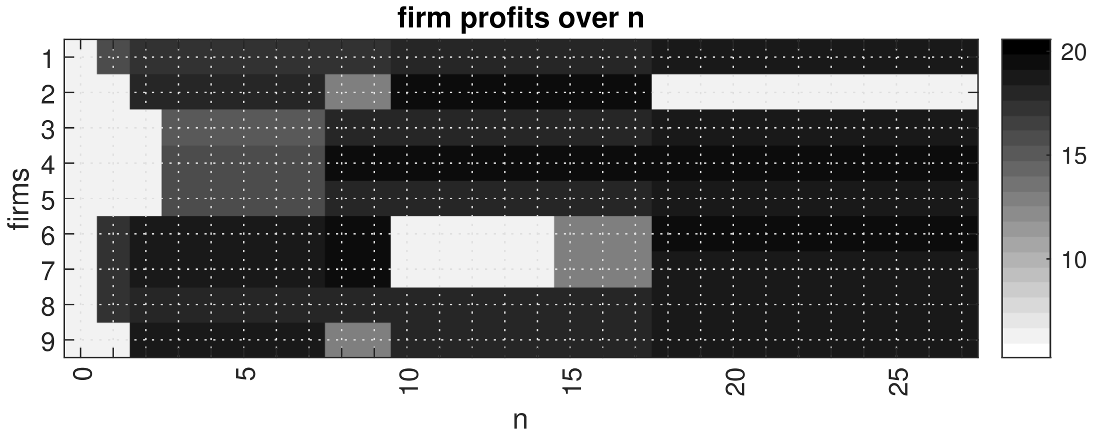


Figure 4.22: Profits with 9 firms, equal parameters, and $\gamma = 1$.

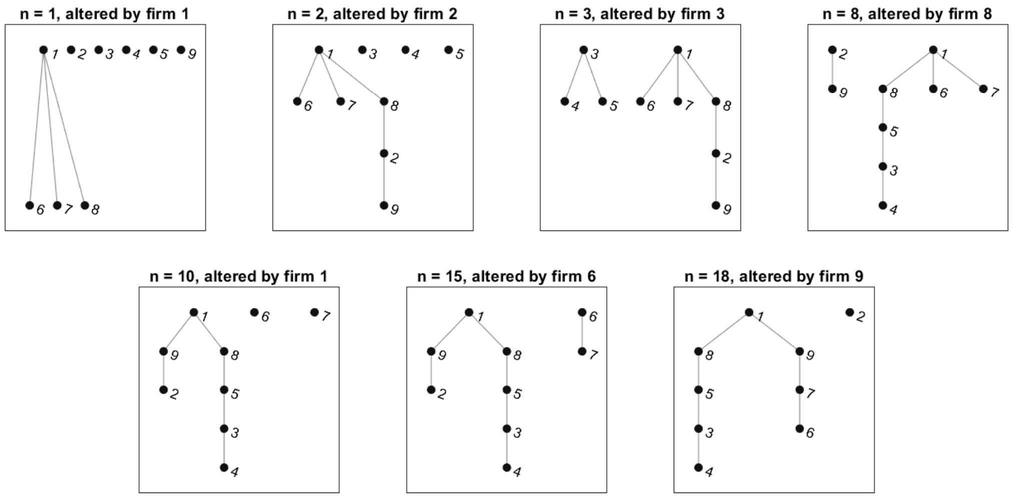


Figure 4.23: Network formation with 9 firms, equal parameters, and $\gamma = 1$.

A smaller γ also results in a different network structure by setting all $\gamma = 1$. We show the resultant network formation in Figures 4.22 and 4.23. Observing both cases shows that both linear and quadratic costs change the network formation, and stable networks are obtained without altering the turn order. Therefore, we demonstrate how our model relates the firms' production cost and profit to the network it is in.

4.4.1.4 Alternate Direct Connection Cost

Next, we start considering the network costs. These costs are the connection cost c and component cost d. The first parameter we consider is $c = 3$. The profits and the network formation can be observed in Figures 4.24 and 4.25. We can see that in this scenario, no firm can ever make two connections. Thus, no system of more than 2 firms can be created.

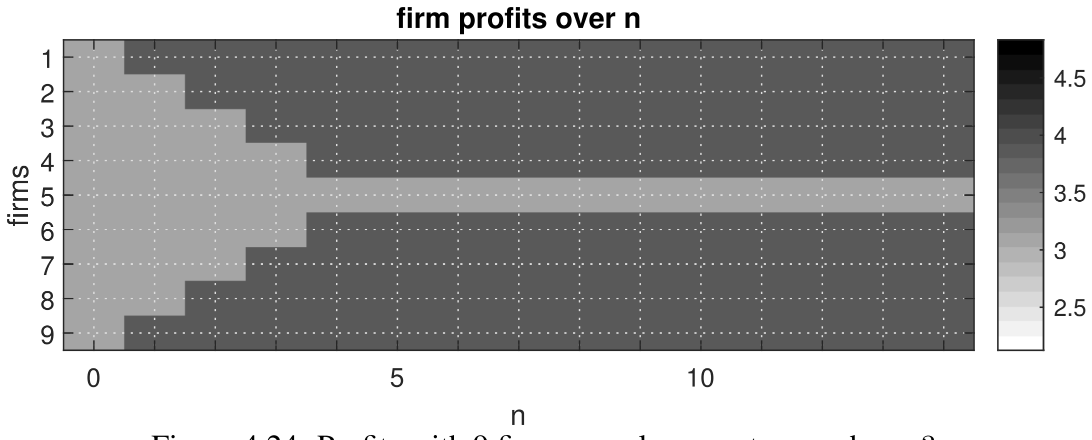


Figure 4.24: Profits with 9 firms, equal parameters, and $c = 3$.

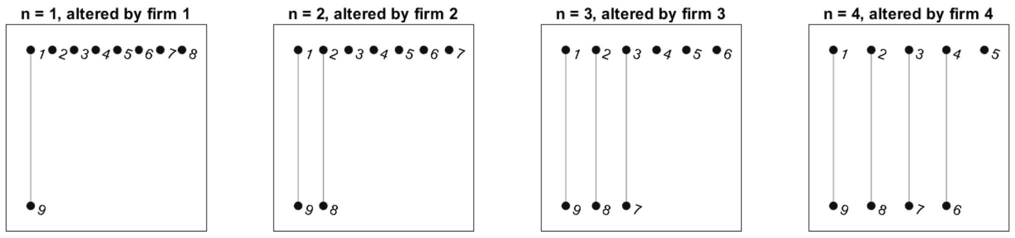


Figure 4.25: Network formation with 9 firms, equal parameters, and $c = 3$.

We follow with another simulation with $c = 0$ to further emphasise its significance to the model. The resultant network progressions with this parameter are given in Figures 4.26 and 4.27. With no cost for direct connections, the first decision-maker connects to 7 other firms. This decision can be attributed to the component cost and diminishing reduction in the production cost. We further see that the left-out firm cannot join the network at the later turns.

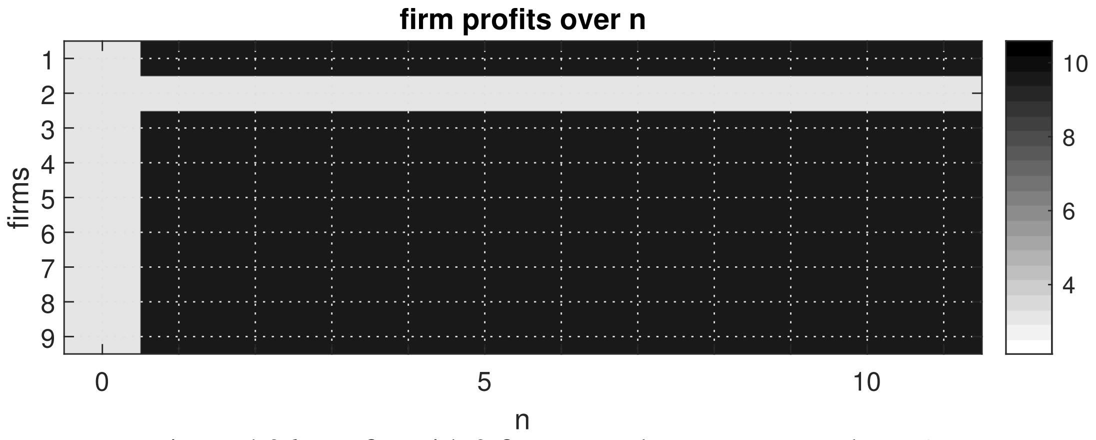


Figure 4.26: Profits with 9 firms, equal parameters, and $c = 0$.

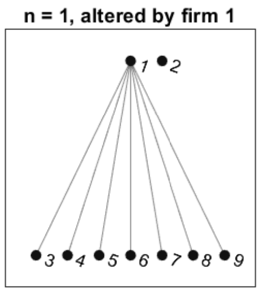


Figure 4.27: Network formation with 9 firms, equal parameters, and $c = 0$.

Further, let us consider a case with $c = 0$ and $d = 0.5$. We see in Figures 4.28 and 4.29 that such a configuration results in smaller components, all formed in a single turn as star networks. We can further see that the star network with firm 1 in its centre is larger. In the first turn, firm 1 is able to connect to four leaves, which creates the best profit for it. Firm 2 is only able to find three leaves to connect, thus creating the second component of the network.

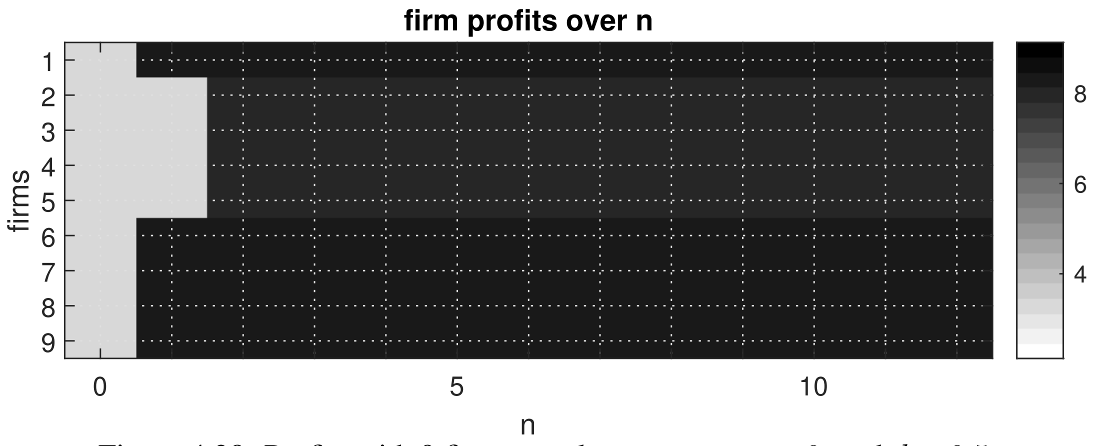


Figure 4.28: Profits with 9 firms equal parameters, $c = 0$, and $d = 0.5$.

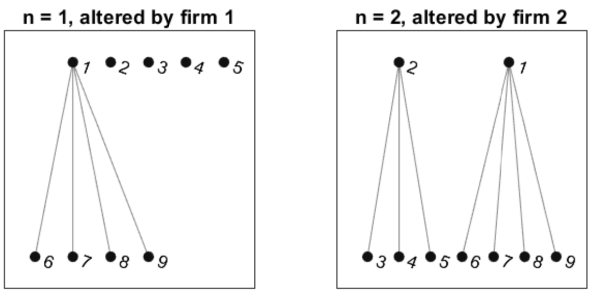


Figure 4.29: Network formation with 9 firms, equal parameters, and $d = 0.5$.

4.4.1.5 Alternate Component Cost

Finally, we examine the component cost. First, we set $d = 1$. Figures 4.30 and 4.31 show that firms cannot afford to make larger connections. Curiously, in the cases of both high c and high d, we end up with firm pairs, but this is the result of different dynamics related to network formation.

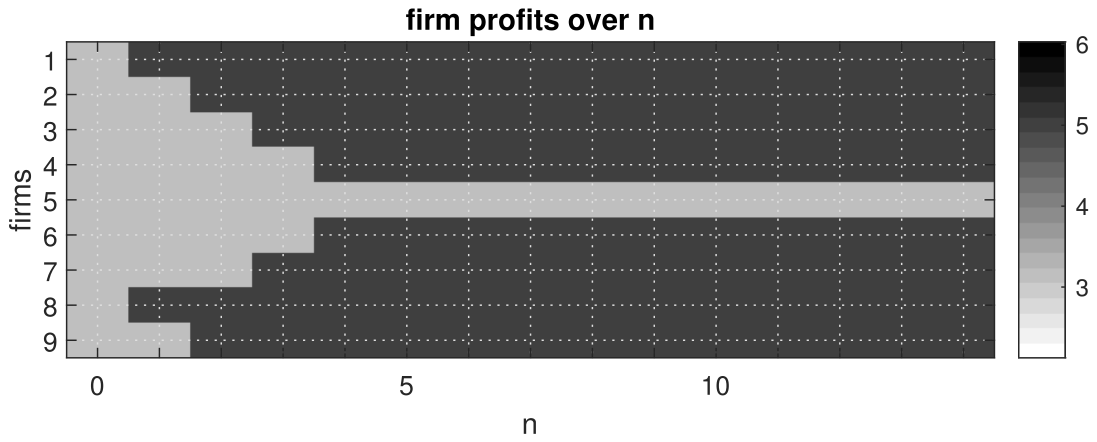


Figure 4.30: Profits with 9 firms, equal parameters, and $d = 1$.

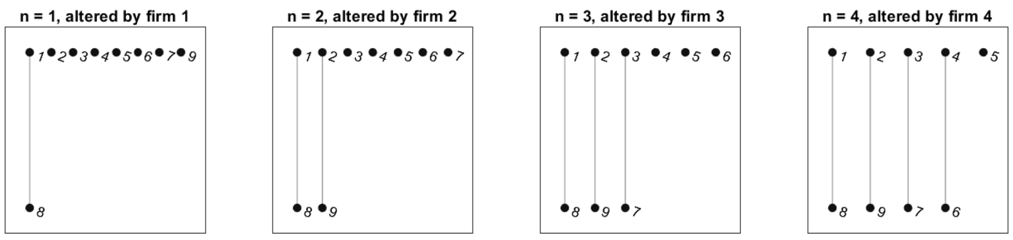


Figure 4.31: Network formation with 9 firms, equal parameters, and $d = 1$.

Next, we can observe the case when $d = 0$. The results of this simulation are given in

Figures 4.32 and 4.33. Without a cost for the component size, we see that firms form a single large component. In this case, stability is reached with firm 3 making four connections.

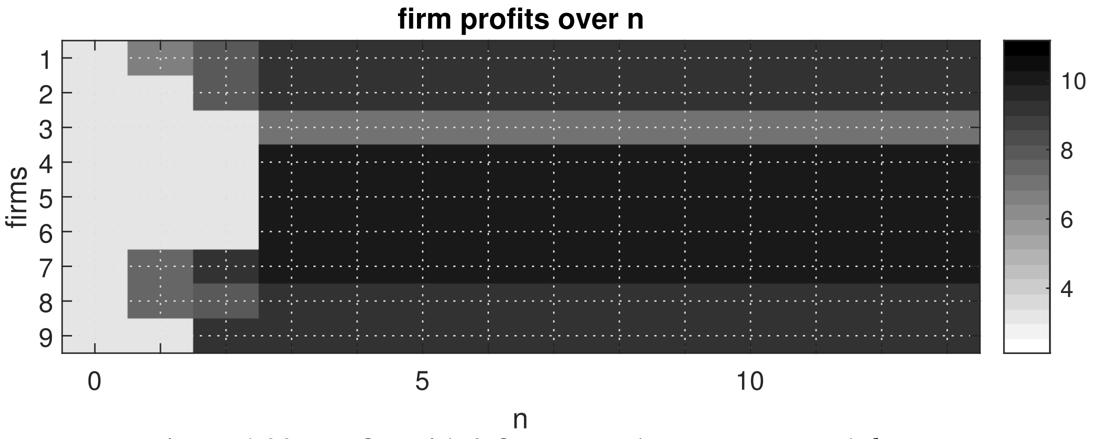


Figure 4.32: Profits with 9 firms, equal parameters, and $d = 0$.

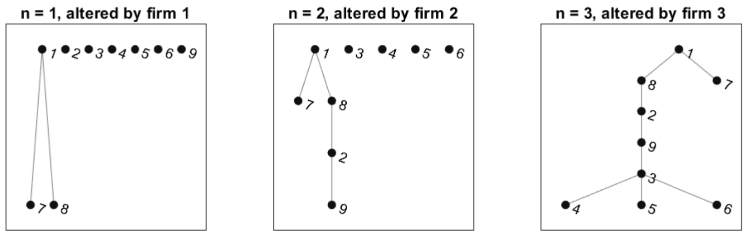


Figure 4.33: Network formation with 9 firms, equal parameters, and $d = 0$.

We finally look at the case with $d = 0$ and $c = 3$ in Figures 4.34 and 4.35. The firms with the largest profit in this configuration are located at the edges of the larger chain. This configuration is only possible because firm 3 connects with both firm pairs, reaching a component size of 5. Since a component of 4 is not large enough to profit from two connections, we see that the two remaining pairs do not connect.

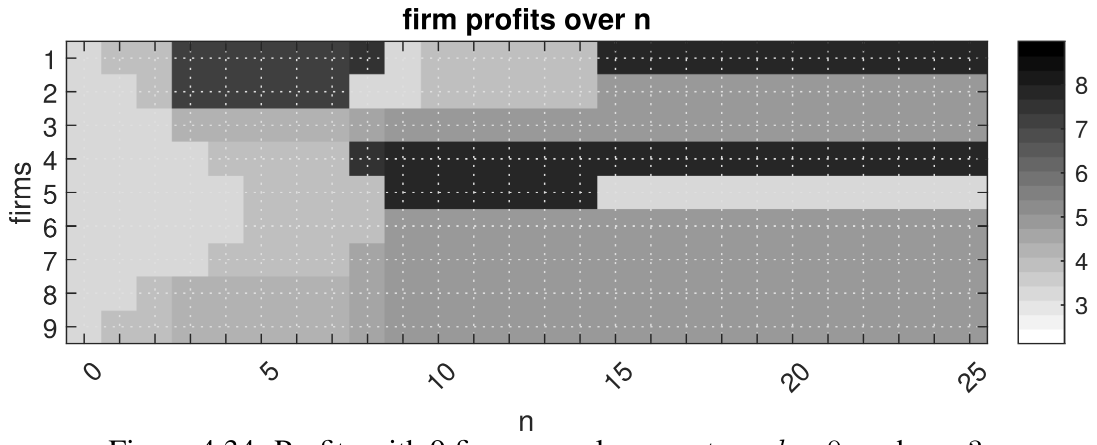


Figure 4.34: Profits with 9 firms, equal parameters, $d = 0$, and $c = 3$.

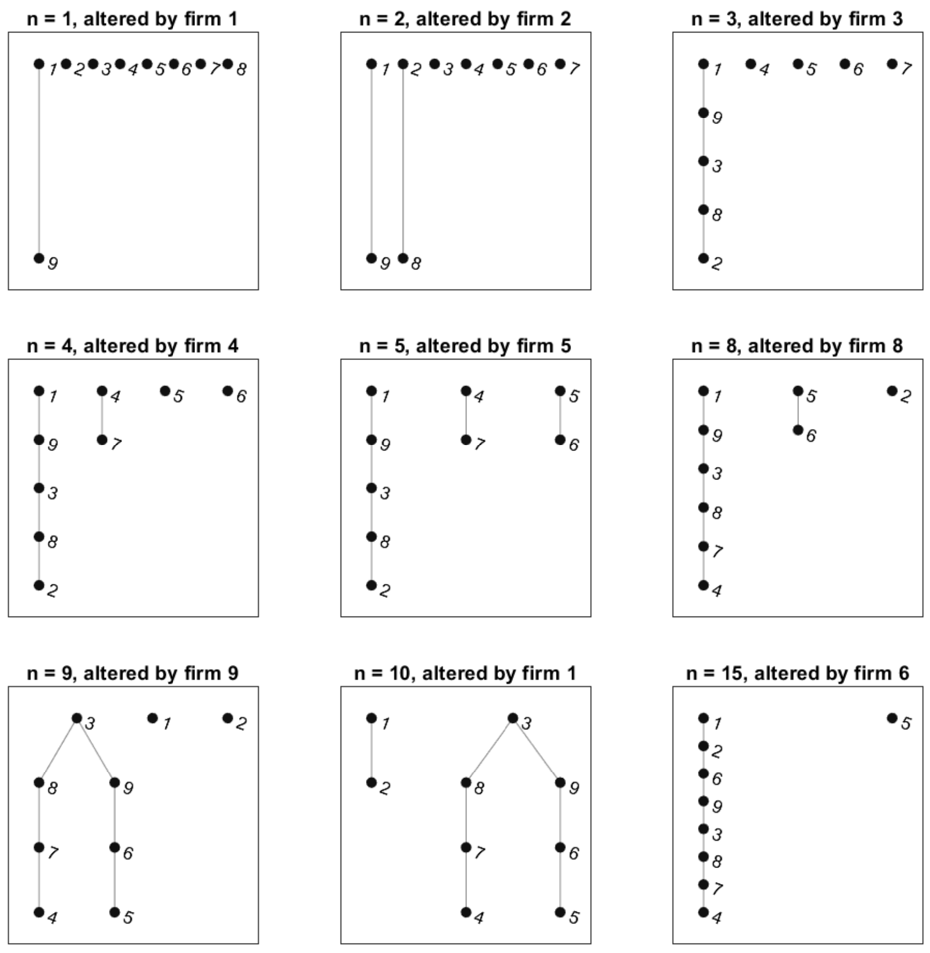


Figure 4.35: Network formation with 9 firms, equal parameters, $d = 0$, and $c = 3$.

4.4.2 Varied Cost

In this section, instead of modelling equal firms, we look at what happens when firms have different characteristics. Matlab script for the 9 firm simulations with equal parameters is in Appendix A.19.

4.4.2.1 Firms of Increasing Linear Cost

We start by setting $\gamma = 2$, $c = 1$, and $d = 0.2$, however for β we have $\beta_i = 2, 2.5, 3 \ldots 6$. This setup offers different incentives for each firm to connect. We expect firms with higher production costs to connect more eagerly, while firms that already produce cheaper would be more subject to network costs while considering connections. Indeed, when we look at Figures 4.36 and 4.37, we see that the two cheapest producers, firms 1 and 2, are located in the shorter chain. And the firms that have higher production costs can all be found in the longer chain.

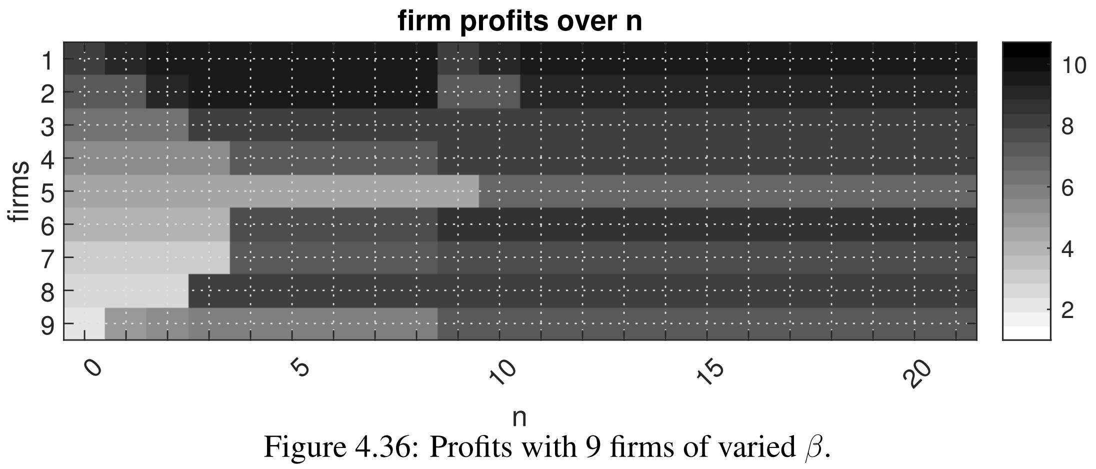


Figure 4.36: Profits with 9 firms of varied β.

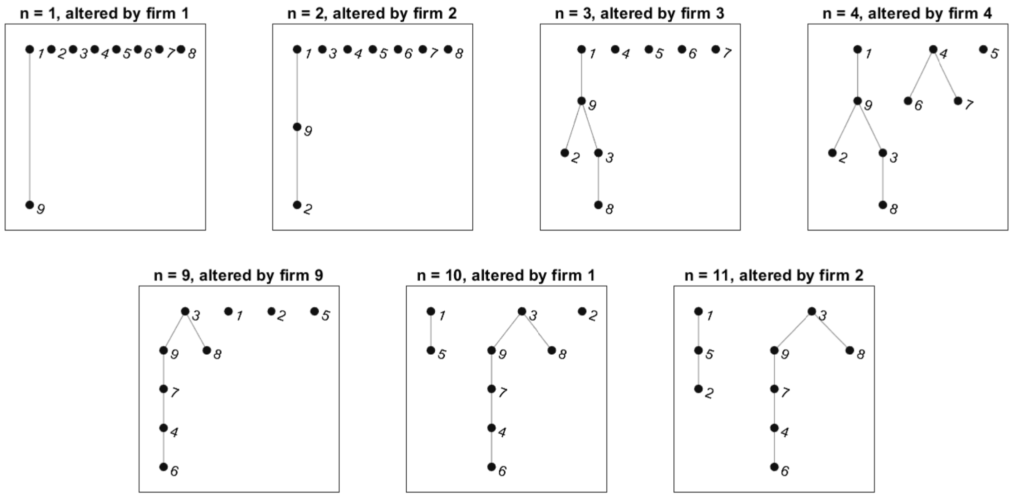


Figure 4.37: Network formation with 9 firms of varied β.

Going further, we shuffle the turn orders with three different seeds. At the varied cost cases, it can be expected that turn order is even less significant for the stable network as firms are already differentiated amongst each other via their production costs. The results are provided in Figures 4.38, 4.39, 4.40, and 4.41. Here, there is a notable difference from the non-shuffled stable network as firm 1 stays alone, thus leading to a stable system of three components. Nevertheless, even that is hardly surprising as the firm already 1 produces as cheaply as firm 9 when it is in a network of 3 firms. Indeed, we can see in all cases that firm 1 profits more than firm 9, the firm in the largest component. Furthermore, once again, firms with higher production costs form larger components, while firms that produce for cheaper are in the smaller component. This result further shows us that the stable networks are the outcomes of the incentive structures introduced in the model.

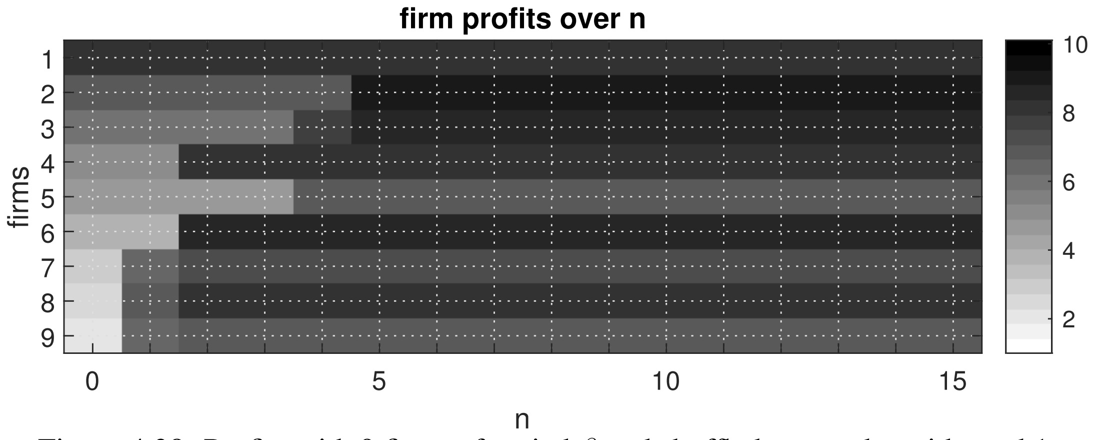


Figure 4.38: Profits with 9 firms of varied β and shuffled turn order with seed 1.

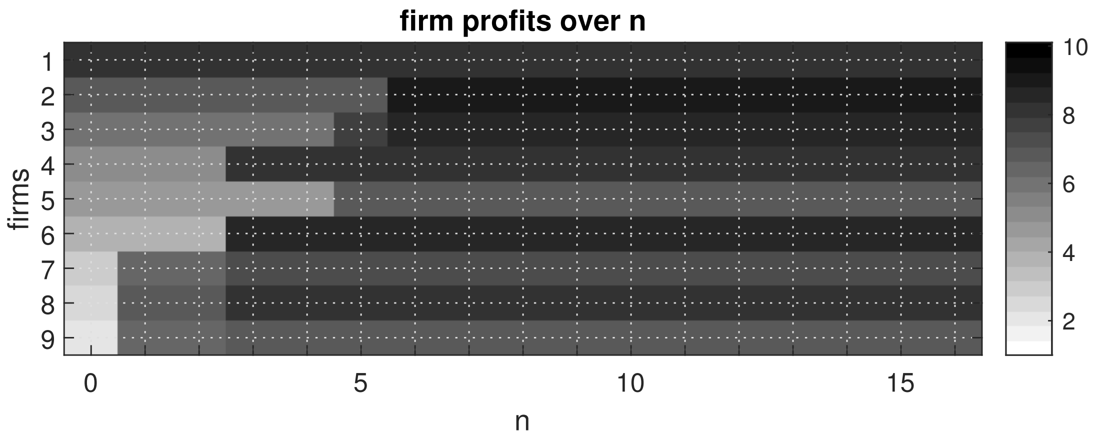


Figure 4.39: Profits with 9 firms of varied β and shuffled turn order with seed 2.

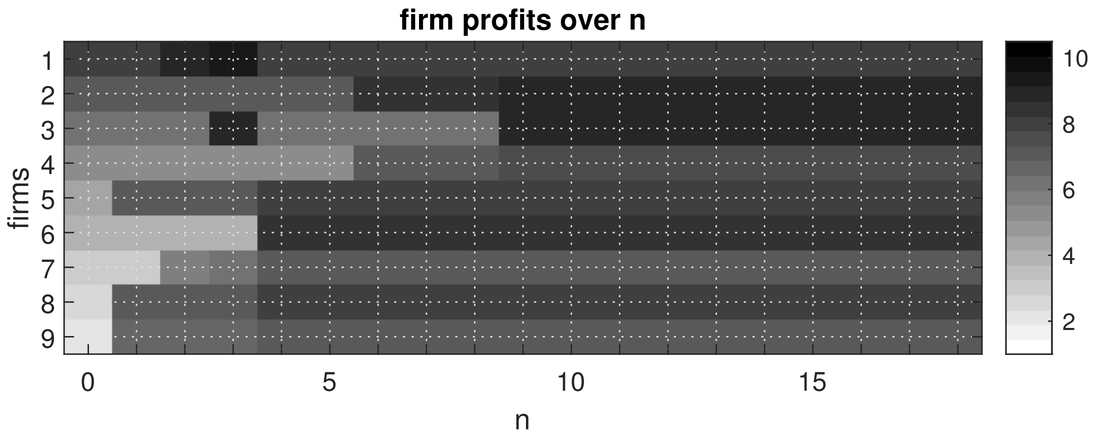


Figure 4.40: Profits with 9 firms of varied β and shuffled turn order with seed 3.

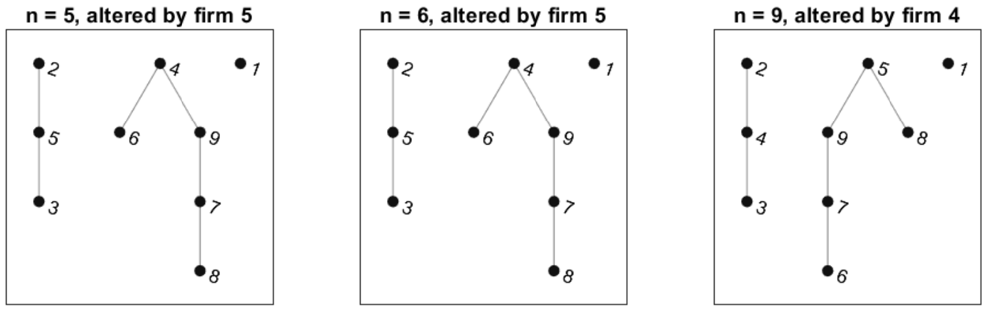


Figure 4.41: Stable networks with 9 firms of varied β and shuffled turn order with seeds 1, 2, 3 from left to right.

4.4.2.2 Firms of Linear Cost Groups

Our findings in the previous section can be further enhanced when we consider a case with four cheap producers at $\beta = 2$, four expensive producers at $\beta = 6$, and one middle firm with $\beta = 4$. This case is simulated once with ordered turns and three shuffled cases, which can be found in Figures 4.42, 4.43, 4.44, 4.45, and 4.46. Here, the incentive to form a network is best observed when we look at the difference between cheap and expensive producers, as the cheap producers form two distinct pairs. In contrast, expensive producers form a single chain at every simulation. We can also see that, while the component sizes in the stable networks are often not dependent on the turn order, firm profits are usually determined by which firms end up at the tips of a chain, which depends on the turn order.

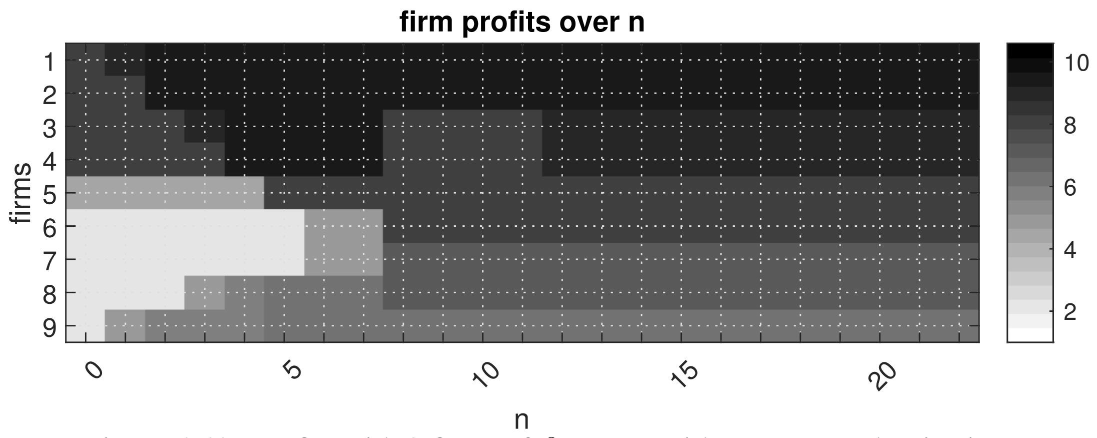


Figure 4.42: Profits with 9 firms of β groups without turn randomisation.

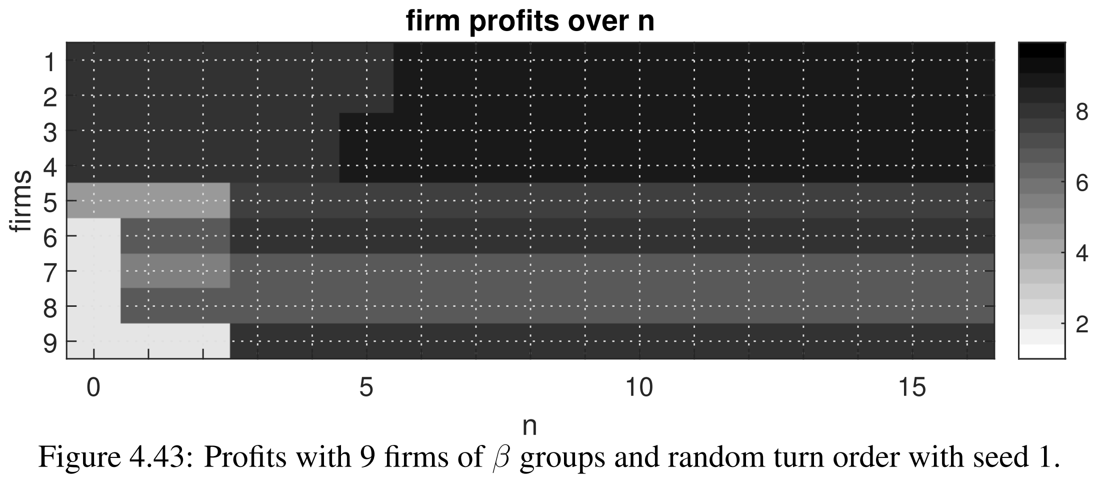


Figure 4.43: Profits with 9 firms of β groups and random turn order with seed 1.

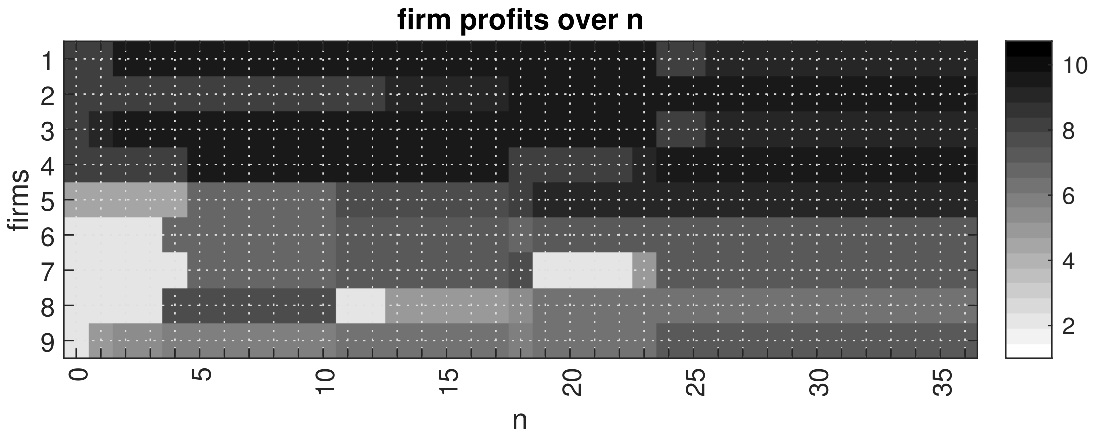


Figure 4.44: Profits with 9 firms of β groups and random turn order with seed 2.

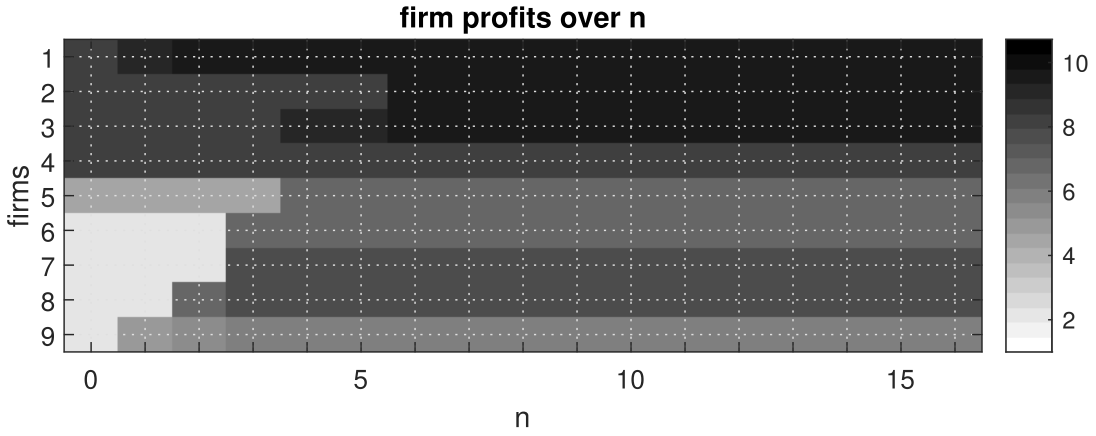


Figure 4.45: Profits with 9 firms of β groups and random turn order with seed 3.

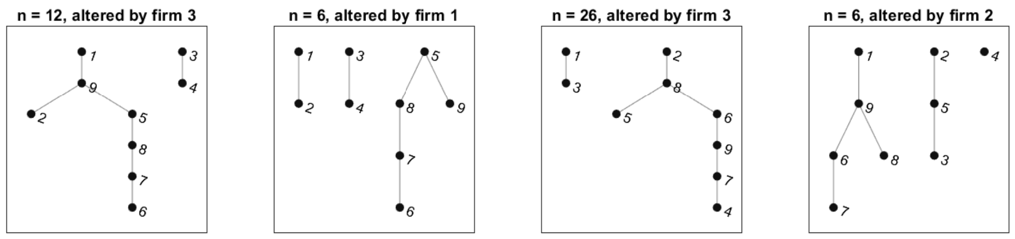


Figure 4.46: Network formation with 9 firms of three β groups, from left to right: non-random, seed 1, seed 2, and seed 3.

We see that a higher linear cost incentivises firms to form larger components, and a higher quadratic cost incentivises firms to form smaller components. We, therefore, can predict that in areas where the cost per production is high, other limitations (that may be expressed by the quadratic cost) are less relevant, and firms cooperate heavily to overcome these costs. In contrast, we predict an already low cost per production to create a lesser incentive for firms to cooperate.

4.5 Sixteen Firms

In this section, in order to better observe upper limits to component sizes imposed by connection parameters, we take $J = 16$. The Matlab script can be found in Appendix A.20.

4.5.1 Equal Parameters

Again we will start with $P = 10$, $\beta = 5$, $\gamma = 2$, $d = 0.2$, $c = 1$. In Figures 4.47 and 4.48, we see that there are four distinct components: three chains of five firms and one unconnected firm. This may lead us to think that in spite of an optimal component size of 8, components with 5 firms are natural to firms of given parameters as none of the three chains can be incentivised to form a chain with 6 members, but when we reconsider the same configuration with a randomised turn order, given in Figures 4.49 and 4.50, we see that three chains can indeed be reached with 16 firms, however, such progression may only happen if the chain to be connected has less than 5 firms in it. The variation in the resultant stable networks can be attributed to a large number of equal firms, as the decision comes down to arbitrarily selecting between choices that offer equal profit.

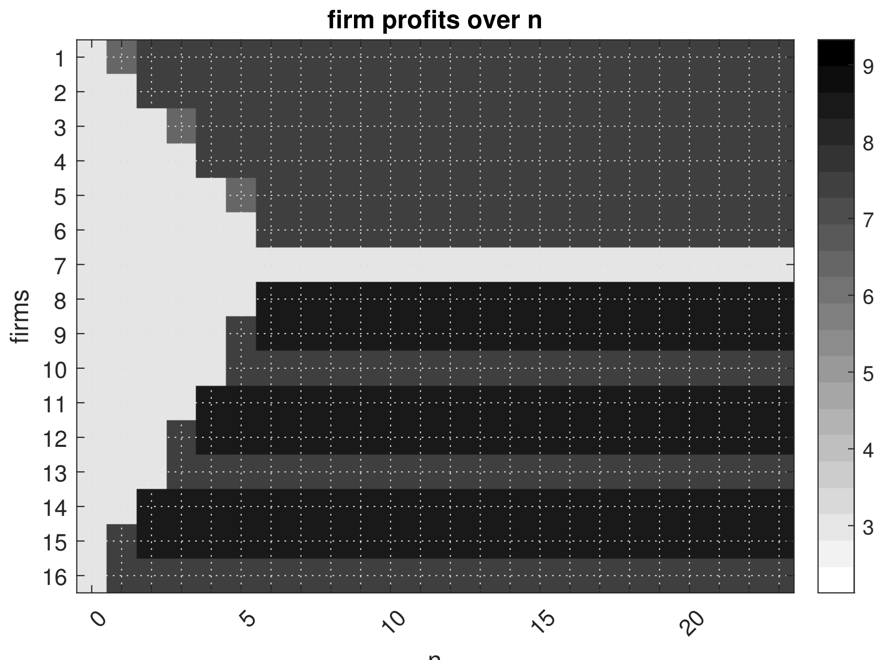


Figure 4.47: Profits with 16 firms and equal parameters.

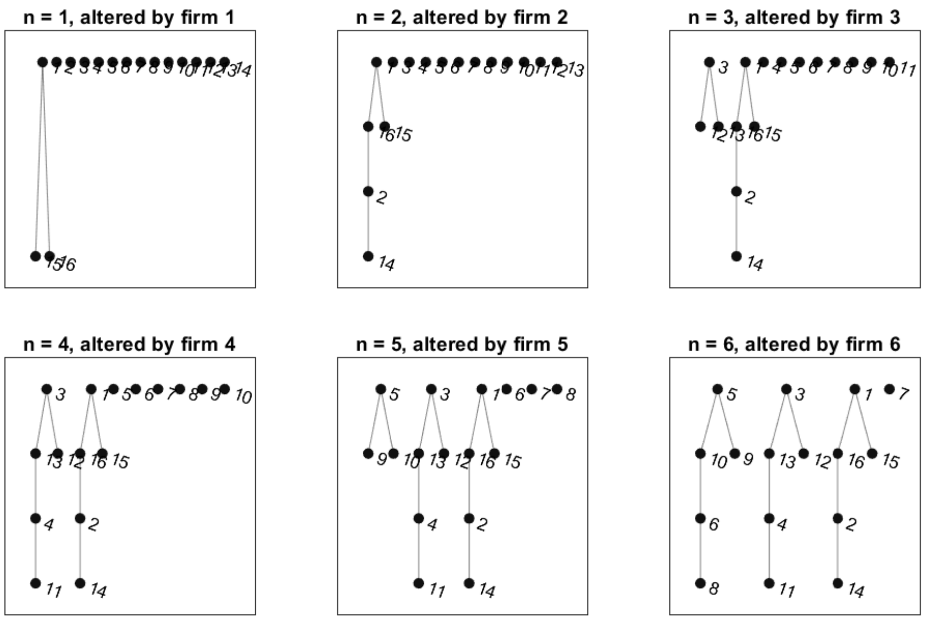


Figure 4.48: Network formation with 16 firms and equal parameters.

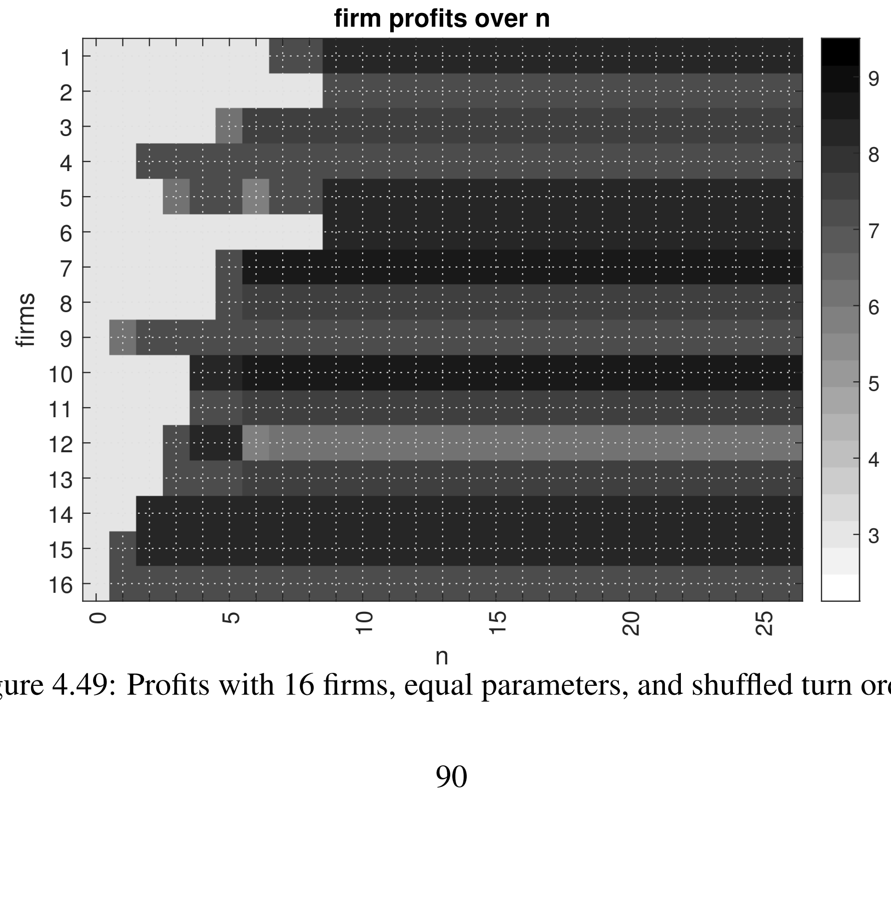


Figure 4.49: Profits with 16 firms, equal parameters, and shuffled turn order.

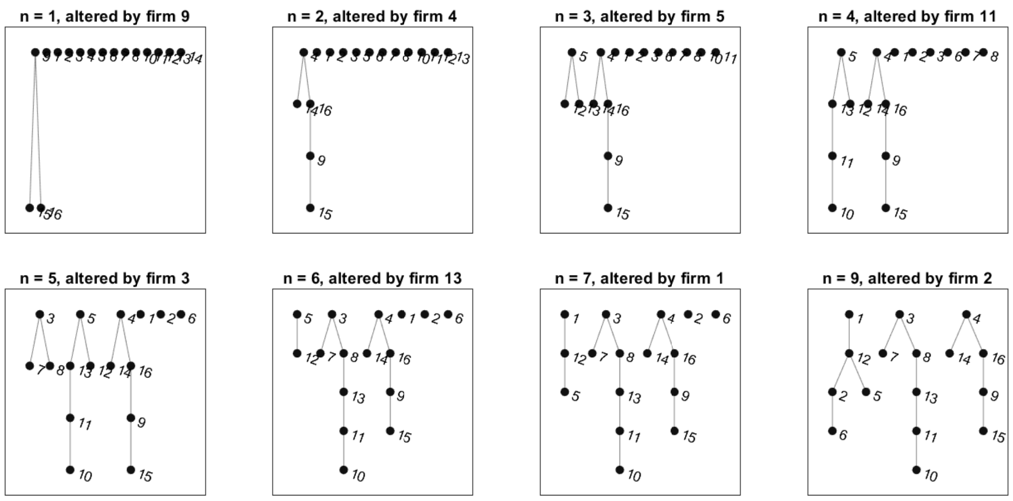


Figure 4.50: Network formation with 16 firms, equal parameters, and shuffled turn order.

4.5.2 Firms of Connection Cost Groups

In this section, we consider the case where we have four different types of firms that vary by connection costs. The exact configuration is given in Table 4.1, and the script for the simulations is provided in Appendix A.20.

Table 4.1: Example firm coefficients.

group	firms	c_i	d_i
A	1,2,3,4	0.5	0.1
B	5,6,7,8	2	0.1
C	9,10,11,12	0.5	0.5
D	13,14,15,16	2	0.5

We make three random order tests with these four groups of players to obtain Figures 4.51, 4.52, 4.53, and 4.54. While the exact formation may take different routes, the resultant systems are fairly similar. However, one notable difference is that we can see a stable system with a non-chain component for the first time. This observation can be attributed to the cheaper connection costs the firm 4 faces. Indeed, in all three trials, a plurality of group A ended up in the larger component. Interestingly, however, we see that firms in group B that face the same cheaper component cost ended up in smaller and larger components. Finally, it is more likely for us to find members of group D in the smaller components due to the general highness of the network costs they face.

Another interesting thing we can observe is that while a firm from group A ends up alone in two of our cases, on average, being in the group A is more profitable. In fact, based on the obtained profits, we can say that $A > C > B > D$ in terms of average profit a firm can obtain. This result is inconclusive in determining the significance of component or direct connection cost. However, we can still see that these values are significant in determining a firm's profit.

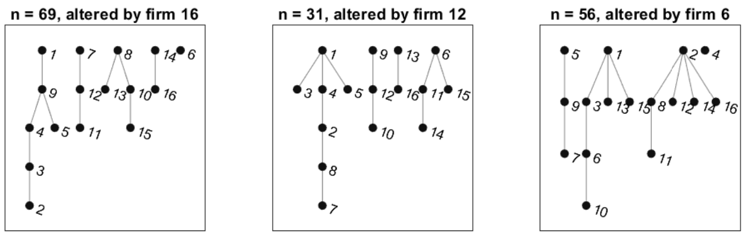

n = 69, altered by firm 16
n = 31, altered by firm 12
n = 56, altered by firm 6

Figure 4.51: Network formation with 16 firms and 4 player groups by seed 1, 2, and 3 from left to right.

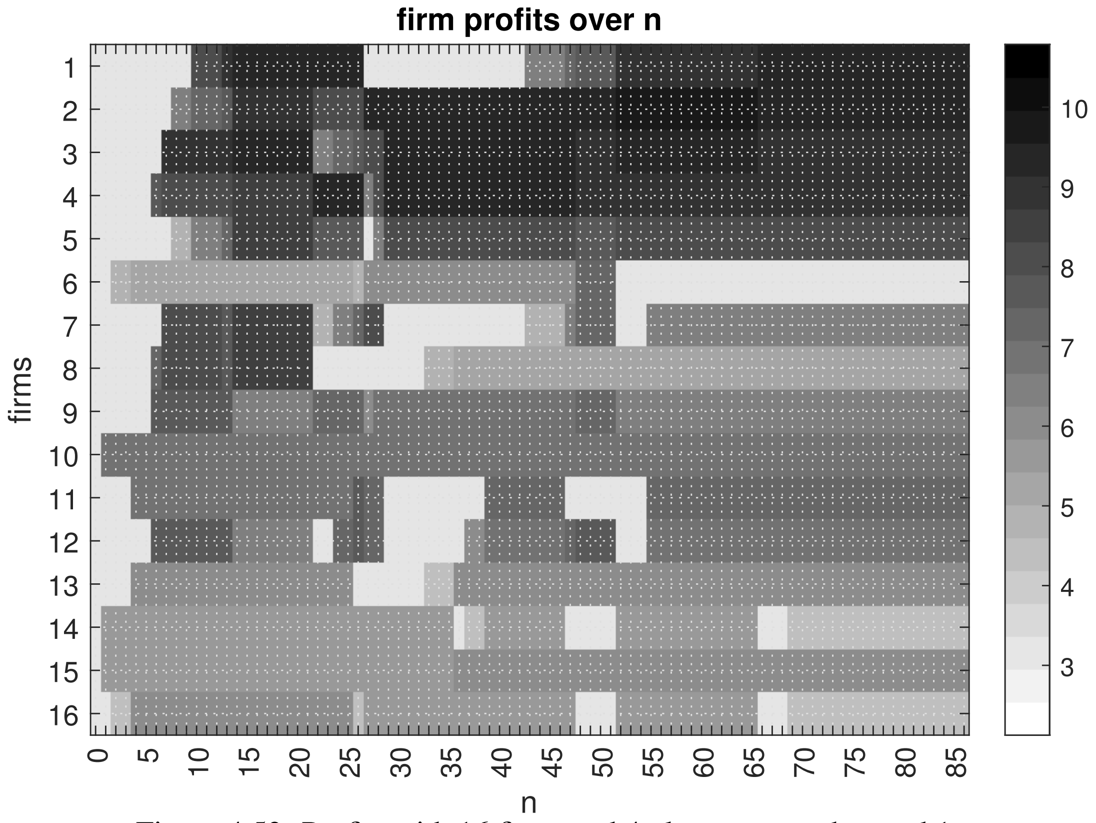


Figure 4.52: Profits with 16 firms and 4 player groups by seed 1.

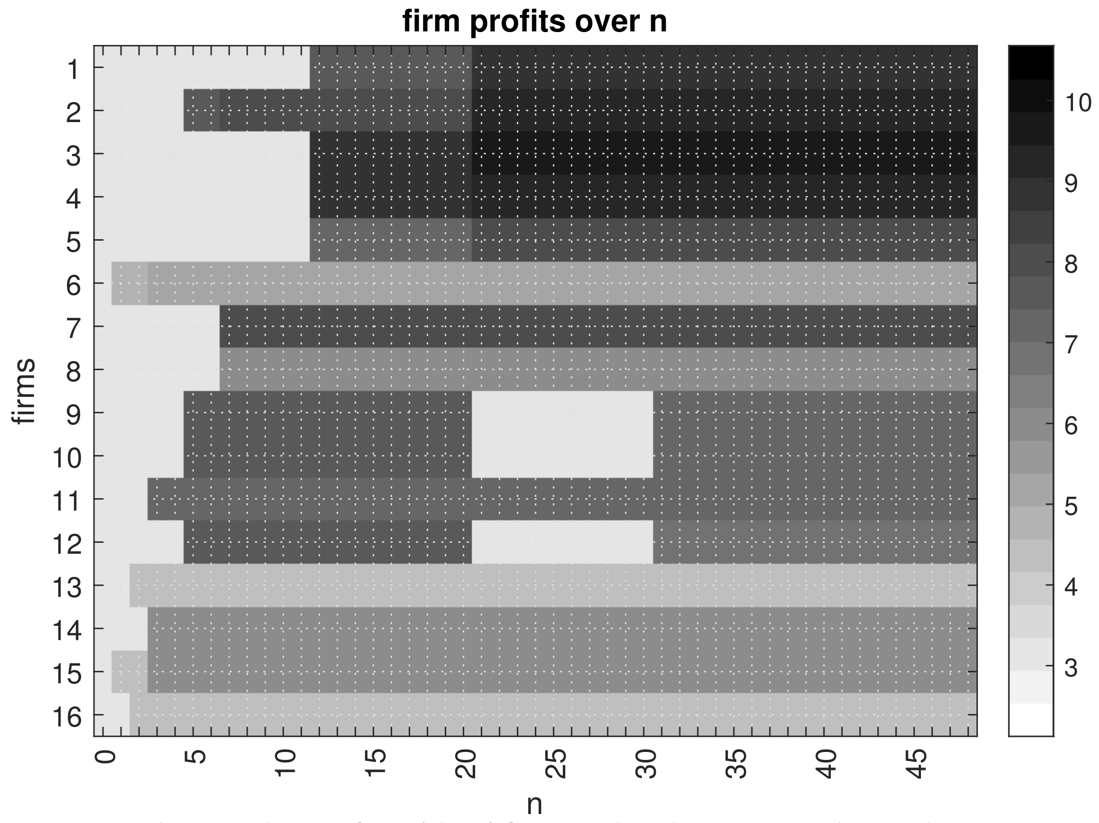


Figure 4.53: Profits with 16 firms and 4 player groups by seed 2.

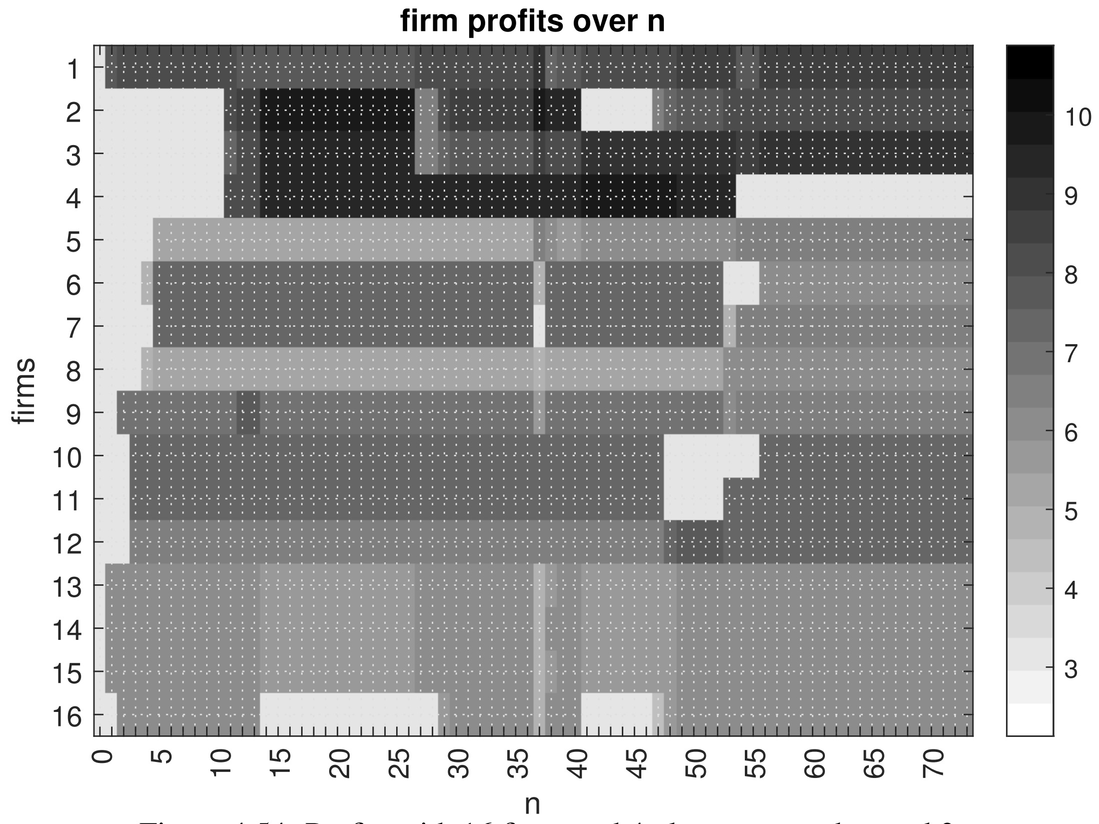


Figure 4.54: Profits with 16 firms and 4 player groups by seed 3.

CHAPTER 5

CONCLUSION

This thesis establishes a specific network formation game and suggests a network stability concept with the multiple connectivity approach. The game is among firms forming networks to reduce their costs and, hence, increase their payoffs. The stable formation of the networks among the firms competing in a market is investigated with the multiple connectivity stability proposed in this thesis as an alternative to the pairwise stability concept developed by Jackson and Wolinsky [31]. The multiple connectivity approach enables individual decision-makers to establish optimal direct network connections by taking into account different network forms simultaneously under certain constraints.

We consider price-taking firms in a market and propose the component profit model, where the firms optimise their profit levels by establishing connections with other firms to lower their production costs. We assume that the firms know all the network forms that can be established, and they can make multiple connections at a time to optimise their profits under certain constraints. When a firm alters its connections in a network, its direct connections consent to the new connection if its profit levels do not decrease. The optimal connection choices when a firm is configuring it's direct connections to other firms are obtained by solving a mixed integer optimisation problem.

Multiple connectivity stability is provided if no firm has an incentive to alter their direct connections in a way that improves their profit levels. That allows the profit-maximising firm to reach an optimal component size based on the model parameters. The problem of computing the optimal number of firms in a component for an arbitrary network is quite complex. We use a linear approximation scheme that allows us to relax the discrete change in the component size due to altering direct connections. Moreover, a well-known mixed integer optimisation technique called the Branch and Bound method is modified to solve our approximated optimisation problem. Furthermore, we develop a procedure to find multiple connectivity stable systems for a larger number of players by applying the multiple connectivity approach to a dynamic network formation game based on pairwise network formation presented by Watts [44].

The simulations with various model parameters reveal the merits of using mixed integer non-linear programming as an optimisation procedure. For a small number of firms, we find that exhaustive searching is the faster option. However, as the number of firms rises, the time consumed by that approach increases exponentially, and the modified branch and bound method works faster.

The simulations give rise to network forms which are multiple connectivity stable but not pairwise stable. Moreover, the multiple connectivity approach results in network forms that can provide higher profits for firms than the pairwise approach. These findings are due to evaluating multiple connections of network forms rather than individual connections. The simulation results reveal the sensitivity of the component size of the networks to the cost parameters related to the production and the network connection.

The multiple connectivity approach and the associated optimisation procedures can be extended to different network formation settings. We aim to provide generalisations and extensions of the multi-connectivity stability proposal in future studies.

REFERENCES

[1] R. J. Aumann and J. H. Dreze, Cooperative games with coalition structures, International Journal of Game Theory, 3, pp. 217–237, 1974.

[2] A. Babus, The formation of financial networks, The RAND Journal of Economics, 47(2), pp. 239–272, 2016.

[3] C. Ballester, A. Calvó-Armengol, and Y. Zenou, Who's who in networks. wanted: The key player, Econometrica, 74(5), pp. 1403–1417, 2006.

[4] P. Baran, On distributed communications networks, IEEE Transactions on Communications Systems, 12(1), pp. 1–9, 1964.

[5] J. A. Barnes, Class and committees in a Norwegian island parish, Human Relations, 7(1), pp. 39–58, 1954.

[6] L. Beal, D. Hill, R. Martin, and J. Hedengren, Gekko optimization suite, Processes, 6(8), p. 106, 2018.

[7] P. Belotti, C. Kirches, S. Leyffer, J. Linderoth, J. Luedtke, and A. Mahajan, Mixed-integer nonlinear optimization, Acta Numerica, 22, p. 1–131, 2013.

[8] J. K. Benson, The interorganizational network as a political economy, Administrative Science Quarterly, 20(2), pp. 229–249, 1975.

[9] F. Bloch and M. O. Jackson, The formation of networks with transfers among players, Journal of Economic Theory, 133(1), pp. 83–110, 2007.

[10] P. Bonacich, Power and centrality: A family of measures, American Journal of Sociology, 92(5), pp. 1170–1182, 1987.

[11] Y. Bramoullé, Anti-coordination and social interactions, Games and Economic Behavior, 58(1), pp. 30–49, 2007.

[12] A. Calvó-Armengol and R. Ilkılıç, Pairwise-stability and Nash equilibria in network formation, International Journal of Game Theory, 38, pp. 51–79, 2009.

[13] S. Cheng, C. Chan, and G. Huang, An integrated multi-criteria decision analysis and inexact mixed integer linear programming approach for solid waste management, Engineering Applications of Artificial Intelligence, 16(5), pp. 543–554, 2003.

[14] J. Clausen, Branch and bound algorithms-principles and examples, Department of Computer Science, University of Copenhagen, pp. 1–30, 1999.

[15] M. M. Cochran and J. A. Brassard, Child development and personal social networks, Child Development, 50(3), pp. 601–616, 1979.

[16] M. Conforti, G. Cornuéjols, and G. Zambelli, *Integer Programming*, pp. 1–44, Springer International Publishing, Cham, 2014.

[17] H. Dawid and T. Hellmann, The evolution of R&D networks, Journal of Economic Behavior & Organization, 105, pp. 158–172, 2014.

[18] M. A. Duran and I. E. Grossmann, An outer-approximation algorithm for a class of mixed-integer nonlinear programs, Mathematical Programming, 36, pp. 307–339, 1986.

[19] B. Dutta and S. Mutuswami, Stable networks, Journal of economic theory, 76(2), pp. 322–344, 1997.

[20] O. Exler, T. Lehmann, and K. Schittkowski, A comparative study of SQP-type algorithms for nonlinear and nonconvex mixed-integer optimization, Mathematical Programming Computation, 4, pp. 383–412, 2012.

[21] L. C. Freeman, Centrality in social networks conceptual clarification, Social Networks, 1(3), pp. 215–239, 1978.

[22] D. Gale and L. S. Shapley, College admissions and the stability of marriage, The American Mathematical Monthly, 69(1), pp. 9–15, 1962.

[23] S. Goyal and S. Joshi, Networks of collaboration in oligopoly, Games and Economic Behavior, 43(1), pp. 57–85, 2003.

[24] C. Gualdani, An econometric model of network formation with an application to board interlocks between firms, Journal of Econometrics, 224(2), pp. 345–370, 2021.

[25] F. Harary and R. Z. Norman, *Graph theory as a mathematical model in social science*, 2, University of Michigan, Institute for Social Research Ann Arbor, 1953.

[26] J. Hirshleifer, From weakest-link to best-shot: The voluntary provision of public goods, Public Choice, 41(3), pp. 371–386, 1983.

[27] M. O. Jackson, *Social and Economic Networks*, Princeton University Press, 2008.

[28] M. O. Jackson and A. van den Nouweland, Strongly stable networks, Games and Economic Behavior, 51(2), pp. 420–444, 2005.

[29] M. O. Jackson and A. Watts, The existence of pairwise stable networks, Seoul Journal of Economics, 14(3), pp. 299–321, 2001.

[30] M. O. Jackson and A. Watts, On the formation of interaction networks in social coordination games, Games and Economic Behavior, 41(2), pp. 265–291, 2002.

[31] M. O. Jackson and A. Wolinsky, A strategic model of social and economic networks, Journal of Economic Theory, 71(1), pp. 44–74, 1996.

[32] M. O. Jackson and L. Yariv, Diffusion of behavior and equilibrium properties in network games, American Economic Review, 97(2), pp. 92–98, 2007.

[33] M. O. Jackson and Y. Zenou, Games on networks, Handbook of game theory with economic applications, volume 4, pp. 95–163, Elsevier, 2015.

[34] L. Katz, A new status index derived from sociometric analysis, Psychometrika, 18(1), pp. 39–43, 1953.

[35] A. Mauleon, J. J. Sempere-Monerris, and V. Vannetelbosch, Networks of knowledge among unionized firms, Canadian Journal of Economics/Revue Canadienne D'économique, 41(3), pp. 971–997, 2008.

[36] A. Mauleon, J. J. Sempere-Monerris, and V. Vannetelbosch, R&D network formation with myopic and farsighted firms, Journal of Economic Behavior & Organization, 208, pp. 203–229, 2023.

[37] R. B. Myerson, Graphs and cooperation in games, Mathematics of Operations Research, 2(3), pp. 225–229, 1977.

[38] A. Najafi and E. W. Richards, Designing a forest road network using mixed integer programming, Croatian Journal of Forest Engineering: Journal for Theory and Application of Forestry Engineering, 34(1), pp. 17–30, 2013.

[39] R. S. Pindyck and D. L. Rubinfeld, *Microeconomics*, Pearson, 2018.

[40] F. R. Pitts, A graph theoretic approach to historical geography, The Professional Geographer, 17(5), pp. 15–20, 1965.

[41] S. Sadic, J. P. de Sousa, and J. A. Crispim, A two-phase milp approach to integrate order, customer and manufacturer characteristics into dynamic manufacturing network formation and operational planning, Expert Systems with Applications, 96, pp. 462–478, 2018.

[42] P. P. Shenoy, On coalition formation: A game-theoretical approach, International Journal of Game Theory, 8(3), pp. 133–164, 1979.

[43] B. Skyrms and R. Pemantle, A dynamic model of social network formation, in *Adaptive Networks: Theory, Models and Applications*, pp. 231–251, Springer, 2009.

[44] A. Watts, A dynamic model of network formation, Games and Economic Behavior, 34(2), pp. 331–341, 2001.

[45] R. J. Wilson, *Introduction to graph theory*, Pearson Education India, 1979.

[46] D. Yue and F. You, Stackelberg-game-based modeling and optimization for supply chain design and operations: A mixed integer bilevel programming framework, Computers & Chemical Engineering, 102, pp. 81–95, 2017.

[47] Z. Zhu, J. Tang, S. Lambotharan, W. H. Chin, and Z. Fan, An integer linear programming and game theory based optimization for demand-side management in smart grid, in *2011 IEEE GLOBECOM Workshops*, pp. 1205–1210, 2011.

APPENDIX A

MATLAB CODES

In this appendix, Matlab codes used are given. Each function's caption includes its purpose and reference to the functions that it uses. Codes are written and run in Matlab R2021a 64-bit in a machine that runs Windows 11.

```
function [new_g] = choice_G(x, G, i)
%CHOICE_G Alters ith row and column of G to convert it to vector x equivalent to \mathcal(G)_i(x,G)
%   x: a vector of binary values
%   G: prior adjacency matrix
%   i: the row and column number
    I = size(G,1);
    new_g = G+0;
    n = 1;
    for k=1:I
        if i==k
            continue
        end
        new_g(i,k) = x(n);
        new_g(k,i) = x(n);
        n = n+1;
    end
end
```

Code A.1: Matlab function equivalent of $\mathcal{G}(x, G)$.

```
function [raw_pi, cluster_matrix, consent_cond, old_profit] = objective_functions(G, i, sp)
%OBJECTIVE_FUNCTIONS Computes objective functions for the choice problem
%   G: prior network matrix
%   i: index of the choice maker
%   sp: system properties struct

    % finds continuous N values
    I = size(G,1);
    G0 = choice_G(zeros(I-1,1),G,i);
    I = size(G0,2);
    N0 =sum(((G0 + eye(I))^I)>0+0,2);
    delta = [];
    n = 1;
    for j = 1:I
        if i==j
            delta = [delta,zeros(I,1)];
            continue
        end
        x = zeros(I-1,1);
        x(n,1) = 1;
        Nx = sum(((choice_G(x,G,i) + eye(I))^I)>0+0,2);
        delta = [delta,Nx-N0];
        n = n + 1;
    end

    % N(x) just for choicemaker
    N_just_i = @(x) 1 + delta(i,:)* (([x(1:i-1);0;x(i:end)]));

    fun_vector = @(x) N0 + (delta*x).*( (delta(i,:)*x)*ones(size(x))  - N0.* (delta*x) +
    ↪  ones(size(x)));
    % N(x) for all firms
    N_fun = @(x) fun_vector([x(1:i-1);0;x(i:end)]);

    single_profit = @(N,D,i) - sp.d(i).*(N-1) - sp.c(i).*D +(N*sp.P -
    ↪  sp.beta(i)).^2./(4*N.^2.*sp.gamma(i)) ;

    % no barrier profit
    raw_pi = @(x)single_profit(N_just_i(x),sum(x,1),i);

    % cluster condition
    G0_M = (G0 + eye(I))^I>0 + 0;
    cluster_matrix = G0_M + 0;
    cluster_matrix(i,:) = [];
    cluster_matrix(:,i) = [];

    NG =sum(((G + eye(I))^I)>0+0,2);
    % consent condition
    profit = @(N,D,i) - sp.d(i).*(N(i)-1) - sp.c(i).*D(i) + (N(i)*sp.P -
    ↪  sp.beta(i)).^2./(4*N(i).^2.*sp.gamma(i)) ;
    old_profit = profit(NG,sum(G,2),(1:I)');

    all_profits = @(x) profit(N_fun(x),sum(choice_G(x,G,i),2),1:size(G,1));
    consent_cond = @(x) consent(x,all_profits,i, old_profit);
end

function [sp] =  consent(x, all_profits, i, current)
    sp1 = all_profits(x)-current;
    sp1(i) = [];
    sp = sp1.*x;
end
```

Code A.2: Optimization problem assembler.

```
function best = exhaustive_searcher(profit, K, CM, consent_cond)
%EXHAUSTIVE_SEARCHER Uses exhaustive search to find the optimal x
%   profit: profit function
%   K: size of the solution vector
%   CM: cluster matrix
%   consent_cond: condition for consent
    li = [];
    for i = 1:2^K
        v = zeros(K,1); n = i-1;
        for k = 1:K
            v(k) = rem(n,2);
            n = fix(n/2);
        end
        n = {};
        n.result = v; n.value = profit(n.result);
        if ~(sum(CM * n.result > 1) || sum(consent_cond(n.result)<0))
            li = [li,n];
        end
    end
    [~, ind] = sort([li.value]);
    li = li(ind);
    best = li(end);
end
```

Code A.3: Uses exhaustive search method to validate branch and bound algorithm.

```
function [obj] = mbb_subproblem_solver(  static_part, x0, pi_fun,  gp)
%MATLAB_SUBPROBLEM_SOLVER Solves subproblem generated by modified branch
%and bound using fmincon, returns a solution struct
%   static_part: the static part decided by earlier steps in B&B
%   x0: initial point of iteration
%   pi_fun: the profit function
%   gp: game properties struct
    obj.static_part = static_part; phi_x = @(x) -1*pi_fun([static_part;x]);
    if size(x0,1)>0
        A = [eye(size(x0,1));-eye(size(x0,1))];  b = [ones(size(x0));zeros(size(x0))];
        obj.variable_part = fmincon(phi_x,x0,A,b,[],[],[],[],[],gp.fmincon_options);
    else
        obj.variable_part = [];
    end
    obj.value = -1 * phi_x(obj.variable_part);
    if size(obj.variable_part,1)<1e-3 % checks if the solution is close enough to integer values
        within_tolerance =true;
    else
        within_tolerance =  sum((obj.variable_part>gp.int_tol)
        ↪  .*((1-obj.variable_part)>gp.int_tol))<1e-3;
    end
    obj.integer_var = obj.variable_part;
    obj.integer_var(obj.integer_var<gp.int_tol) =0;
    obj.integer_var((1-obj.integer_var)<gp.int_tol) =1;
    if within_tolerance % values are readjusted to integer values
        obj.done = true; obj.variable_part = obj.integer_var;
        obj.value = -1* phi_x(obj.variable_part);
    else
        obj.done = false;
    end
    obj.result = [obj.static_part;obj.variable_part];
end

```

Code A.4: Solves the sub-problems generated by the modified branch and bound algorithm. Uses fmincon.

```
function [largest] = modified_branch_and_bound( phi, pruner, x0, gp,  cluster_matrix, consent_cond)
%MODIFIED_BRANCH_AND_BOUND Applies modified b&b algorithm to solve choice problem,
%   returns one of the largest profitting solution
%   phi: objective function
%   pruner: profit from previous G for pruning
%   x0: initial choice,
%   gp: game properties struct
%   cluster_matrix: an IxI matrix such that cm*x <= 1
%   consent_cond: a function that returns a vector that takes negative
%   values if a particualr firm does not consent to the connection
    first_subsolution = mbb_subproblem_solver([],x0,phi,gp );
    active_list = [first_subsolution];

    all_zeros = mbb_subproblem_solver(x0,[],phi,gp);
    inactive_list = [all_zeros];

    max_v = max(all_zeros.value,pruner);
    for  i = 1:2^size(x0,1)
        if size(inactive_list,1) > 0 %pruning
            active_list = active_list(([active_list.value]+1e-3)>= max_v);
        end

        if size(active_list,1)==0 % evaluation_over
            break
        end
        largest = active_list(end);

        % split to subproblems
        s1 = [largest.static_part;1];
        s0 = [largest.static_part;0];
        l1 = [];
        l2 = [];
        for this_s = [s0,s1]
            this_x0 = x0((size(this_s,1)+1):end);
            nn = [this_s;zeros(size(this_x0))];

            if sum(cluster_matrix * nn > 1)  % if cluster fails by only the static values
                continue
            end

            bbs1 = mbb_subproblem_solver(this_s,this_x0,phi,gp);
            if bbs1.done % if an integer solution is reached
                res = bbs1.result;
                if sum(cluster_matrix * res > 1) || sum(consent_cond(res)<0) % and if conditions not
                ↪  are met
                    if size(bbs1.variable_part,1)>0 % if variable part is not finished reconsider
                        bbs1.done = false;
                        l1 = [l1;bbs1];
                    end
                else % if conditions are met
                    l2 = [l2;bbs1]; % add to the done list
                    if bbs1.value > max_v % and save the value for pruning
                        max_v = bbs1.value;
                    end

                end
            else
                l1 = [l1;bbs1];
            end
        end
        active_list = [active_list(1:(end-1));l1];
        inactive_list = [inactive_list;l2];
    end
    [~, ind] = sort([inactive_list.value]);
    inactive_list = inactive_list(ind);
    largest = inactive_list(end);
end
```

Code A.5: Implementation of the Modified Branch and Bound algorithm as given in Section 3. Uses A.4.

```
function [obj] = pure_subproblem_solver( static_part, x0, phi,  gp, small_G0, consent_cond)
%PURE_SUBPROBLEM_SOLVER Solves subproblem generated by pure branch
%and bound using fmincon with non linear conditions,
%returns a solution struct
%   static_part: the static part decided by earlier steps in B&B
%   x0: initial point of iteration
%   pi_fun: the profit function
%   gp: game properties struct
%   cluster_matrix: an IxI matrix such that cm*x <= 1
%   consent_cond: a function that returns a vector that takes negative
%   values if a particualr firm does not consent to the connection
    obj.static_part = static_part;
    phi_x = @(x) -1*phi([static_part;x]);
    smaller_lb = size(static_part,1);
    obj.invalid = false;
    if size(x0,1)>0
        smaller_G0 = small_G0(smaller_lb+1:end,smaller_lb+1:end);
        A = [eye(size(x0,1));-eye(size(x0,1));smaller_G0];
        b = [ones(size(x0));zeros(size(x0));ones(size(x0))];
        c = @(x) c_wrap(consent_cond, static_part, x);
        [obj.variable_part ,~,exitflag,~] = fmincon(phi_x,x0,A,b,[],[],[],[],c,gp.fmincon_options);
        if exitflag~=1
            obj.invalid=true;
        end
    else
        obj.variable_part = [];
        if sum(small_G0 * static_part > 1) || sum(consent_cond(static_part)<0)
           obj.invalid = true;
        end
    end
    obj.value = -1 * phi_x(obj.variable_part);

    if size(obj.variable_part,1)<1e-3 % checks if the solution is close enough to integer values
        within_tolerance =true;
    else
        within_tolerance =  sum((obj.variable_part>gp.int_tol)
        ↪  .*((1-obj.variable_part)>gp.int_tol))<1e-3;
    end

    obj.integer_var = obj.variable_part;
    obj.integer_var(obj.integer_var<gp.int_tol) =0;
    obj.integer_var((1-obj.integer_var)<gp.int_tol) =1;

    if within_tolerance % values are readjusted to integer values
        obj.done = true;
        obj.variable_part = obj.integer_var;
        obj.value = -1* phi_x(obj.variable_part);
        res = [obj.static_part;obj.variable_part];
        if sum(consent_cond(res)<0) || sum(small_G0 * res > 1)
           obj.invalid = true;
        end

    else
        obj.done = false;
    end

    obj.result = [obj.static_part;obj.variable_part];

end

function [c,ceq] = c_wrap(consent_cond, static_part, x)
    c =  -1*consent_cond([static_part;x]);
    ceq = [];
end
```

Code A.6: Solves the sub-problems generated by the pure branch and bound algorithm. Uses *fmincon*.

```
function [largest] = pure_branch_and_bound( phi,  pruner, x0, gp, cluster_matrix, consent_cond )
%MODIFIED_BRANCH_AND_BOUND Applies pure b&b algorithm to solve choice problem,
%   returns one of the largest profitting solution
%   phi: objective function
%   pruner: profit from previous G for pruning
%   x0: initial choice,
%   gp: game properties struct
%   cluster_matrix: an IxI matrix such that cm*x <= 1
%   consent_cond: a function that returns a vector that takes negative
%   values if a particualr firm does not consent to the connection
    first_subsolution = pure_subproblem_solver([],x0,phi,gp, cluster_matrix, consent_cond );
    active_list = [first_subsolution];

    all_zeros = pure_subproblem_solver(zeros(size(x0)),[],phi,gp, cluster_matrix, consent_cond);
    inactive_list = [all_zeros];

    max_v = max(all_zeros.value,pruner);
    for  i = 1:2^size(x0,1)
        if size(inactive_list,1) > 0 %pruning
            active_list = active_list(([active_list.value]+1e-3)>= max_v);
        end

        if size(active_list,1)==0
            % evaluation_over
            break
        end
        largest = active_list(end);

        % split to subproblems
        s1 = [largest.static_part;1];
        s0 = [largest.static_part;0];
        l1 = [];
        l2 = [];
        for this_s = [s0,s1]
            this_x0 = x0((size(this_s,1)+1):end);
            bbs1 = pure_subproblem_solver(this_s,this_x0,phi,gp,cluster_matrix, consent_cond);
            if bbs1.invalid
                if size(bbs1.variable_part,1)>0
                    bbs1.done = false;
                    l1 = [l1;bbs1];
                end
            else
                if bbs1.done % if an integer solution is reached
                    l2 = [l2;bbs1]; % add to the done list
                    if bbs1.value > max_v % and save the value for pruning
                        max_v = bbs1.value;
                    end
                else
                    l1 = [l1;bbs1];
                end
            end
        end
        active_list = [active_list(1:(end-1));l1];
        inactive_list = [inactive_list;l2];
    end

    [~, ind] = sort([inactive_list.value]);
    inactive_list = inactive_list(ind);
    largest = inactive_list(end);
end

```

Code A.7: Implementation of the Pure Branch and Bound algorithm. Uses A.6.

```
function [records] = gamer(sp,gp, random_order)
%GAMER The function that executes stable system finding procedure with given system properties
%   sp: system properties struct with, P, beta, gamma, c, d
%   gp: game properties struct
%   random_order: if true, turn order is randomized between turns
    G = zeros(size(sp.beta,1)); % empty matrix infered from sp
    I = size(G,1); % I inferred from
    profit = @(N,D,i) (N(i)*sp.P - sp.beta(i)).^2./(4*N(i).^2.*sp.gamma(i)) - sp.d(i).*(N(i)-1) -
    ↪  sp.c(i).*D(i);
    C = 20;
    n = 0;
    records = recorder(true); % initialize the recording
    N0 = ones(I,1);
    D0 = zeros(I,1);
    start_profit = profit(N0,D0,1:size(G,1));
    tr = turn_record(G, N0, D0, start_profit,0);
    records= records.add_turn_record(tr);
    samer = 0; % number of turns that G is not changed
    for cycle = 1:C
        old_G = G;
        order = (randperm(length(1:I)));
        for i_1 = 1:I
            if random_order
                i = order(i_1);
            else
                i = i_1;
            end
            x0 = [G(i,1:(i-1)),G(i,(i+1):end)]';
            [pi_fun, CM, consent, baseline] = objective_functions(G, i, sp); % calculate the move
            ↪
            largest_phi_m = modified_branch_and_bound( pi_fun, baseline(i), x0, gp, CM, consent);
            xn_m = largest_phi_m.result;
            val_m  =largest_phi_m.value;
            if gp.pure_bnb_check
                largest_phi_p = pure_branch_and_bound( pi_fun, baseline(i), x0, gp, CM, consent);
                if (norm(xn_m - largest_phi_p.result)>1e-3) &&
                ↪  (norm(val_m-largest_phi_p.value)>1e-3)...
                        && (baseline< max(val_m,largest_phi_p.value))
                    fprintf("discrepency with pure bnb \n");
                end
            end
            if gp.exhaustive_check
                largest_ind = exhaustive_searcher(pi_fun,I-1,CM, consent);
                if norm(val_m - largest_ind.value)>1e-3 && (val_m> baseline(i)+1e-6)
                    fprintf("%d disc at x %s %s\n", n,mat2str(largest_ind.result),mat2str(xn_m))
                end
            end
            if baseline(i)+1e-6<val_m % firm profits from altering the network
                Gn = choice_G(xn_m,G,i); % network is altered
            else
                Gn = G; % it choses the old order
            end
            n = n+1;
            Dn = sum(Gn,2);
            Nn = sum(((Gn + eye(I))^I)>0+0,2);
            tr = turn_record(Gn,Nn,Dn,  profit(Nn,Dn,1:size(G,1)),i);
            records= records.add_turn_record(tr);
            if( norm(G-Gn)<1e-3)
                samer  = samer+1;
            else
                samer = 0;
            end
            G = Gn;
        end
        if samer>=I
            records = records.truncate(samer-I);
            disp(sprintf("true in %d",n-I))
            break
        end
        if norm(G-old_G)<1e-3
            disp("cyclic")
            break
        end
    end
end
```

Code A.8: Finds stable systems for MCG as given in Section 4. Uses A.14, A.15, A.2, A.5, A.7, A.3, A.1

```
function [real_vals, best_vals] = check_stability(G, sp, gp)
%CHECK_STABILITY Checks if G is MC stable  returns best and real vals,
% if any best val is larger than corr. real val, system is not stable
%   G: a network to be checked
%   sp: firm parameters, gp: game parameters
    J= size(G,1); best_vals = zeros(J,1); real_vals = zeros(J,1);
    for i = 1:size(G,1)
        [pi_fun, CM, consent,baseline] = objective_functions(G, i, sp); % calculate the move
        ↪
        result = exhaustive_searcher(pi_fun,size(G,1)-1,CM, consent);
        val_m = result.value; best_vals(i) = result.value; real_vals(i) = baseline(i);
        if gp.mod_bnb_check
            x0 = [G(i,1:(i-1)),G(i,(i+1):end)]';
            largest_phi_m = modified_branch_and_bound( pi_fun, baseline(i), x0, gp, CM, consent);
            val_mm  =largest_phi_m.value;
            if val_m>val_mm
                disp("problem 1")
            elseif val_mm>val_m
                disp("problem 2")
            end
        end
    end
end

```

Code A.9: Checks if a system is MC stable.

```
function [adders,breakers, baseline] = pw_checker(G, sp)
%PW_CHECKER Checks if G is PW stable according to the parameters in sp
%returns best and real vals, if any best val is larger than corr. real val,
%system is not stable
%   G: a network to be checked
%   sp: firm parameters
    profit = @(N,D,i) - sp.d(i).*(N-1) - sp.c(i).*D +(N*sp.P - sp.beta(i)).^2./(4*N.^2.*sp.gamma(i))
    ↪  ;
    J = size(G,1);
    D = sum(G,2); N = sum(((G + eye(J))^J)>0+0,2);
    baseline = profit(N,D,1:J); pairs = nchoosek(1:J,2);
    change = 0; breakers = []; adders = [];
    for pindex = 1:size(pairs,1)
        i = pairs(pindex,1); j = pairs(pindex,2);
        G_1 = set_connection(G,i,j,1); G_0 = set_connection(G,i,j,0);
        D_1 = sum(G_1,2); N_1 = sum(((G_1 + eye(J))^J)>0+0,2);
        D_0 = sum(G_0,2); N_0 = sum(((G_0 + eye(J))^J)>0+0,2);
        consent_i = profit(N_1(i),D_1(i),i) - profit(N_0(i),D_0(i),i);
        consent_j = profit(N_1(j),D_1(j),j) - profit(N_0(j),D_0(j),j);
        if (consent_i<0)||(consent_j<0)
            difference = G(i,j) == 1;
        elseif (consent_i>0)||(consent_j>0)
            difference = G(i,j) == 0;
        else
            difference = G(i,j) == 1;
        end
        if difference
            if G(i,j)
                breakers(end+1,1:2) = [i,j];
            else
                adders(end+1,1:2) = [i,j];
            end
            change = change+1;
        end
    end
end

function [new_G] = set_connection(G,i,j,val)
    new_G = G+0;
    new_G(i,j) = val;
  new_G(j,i) = val;
end
```

Code A.10: Checks if a system is PW stable.

```
function [] = process_mc_stability(a1, sp, gp, J, path)
%PROCESS_MC_STABILITY, checks all given networks to find stable ones, and
%plots the results
%   a1: list of networks to be checked, must have JxJ graphs
%   sp: system parameters, gp: game parameters (used if MBB is used )
%   J: number of elements in
%   path: path to be recorded
    p1 = plot(graph(ones(J,J)-eye(J)));
    xdat = p1.XData; ydat = p1.YData;
    close();
    stab_text = ["s","u"]; m= ["o","^","v"]; su = [1,1];
    for i = 1:size(a1,2)
        G = a1{i};
        [real_vals, best_vals] = check_stability(G,sp,gp);
        us = any(best_vals>real_vals);
        f = figure;
        p = plot(graph(G));
        p.NodeLabel = compose("%d (%.2f)",(1:J)', real_vals);
        p.Marker = m(((best_vals>real_vals)+2*(best_vals<real_vals))+1);
        p.MarkerSize = 6; p.XData = xdat; p.YData = ydat;
        hold on
        f.Position = [100 100 300 300];
        saveas(f,path + sprintf("%s_iso_%d.eps",stab_text(us+1),su(us+1)),'epsc');
        close(); su(us+1) = su(us+1) + 1;
    end
end
```

Code A.11: Processes MC stability of given graphs and plots results. Uses A.9.

```
function [] = process_pw_stability(a1,sp,J,path)
%PROCESS_PW_STABILITY, checks all given networks to find stable ones, and
%plots the results
%   a1: list of networks to be checked, must have JxJ graphs
%   sp: system parameters
%   J: number of elements in
%   path: path to be recorded
    p1 = plot(graph(ones(J,J)-eye(J)));
    xdat = p1.XData; ydat = p1.YData;
    close();
    stab_text = ["s","u"]; su = [1,1];
    for i = 1:size(a1,2)
        G = a1{i};
        [adders,breakers,real_vals] = pw_checker(G,sp); %checks stability, rest is visual
        us = (size(adders,1) + size(breakers,1))>0;
        f = figure;
        p = plot(graph(G));
        p.XData = xdat; p.YData = ydat; p.LineWidth = 3;
        p.NodeLabel = compose("%d (%.2f)",(1:J)', real_vals);
        G2 = zeros(size(G));
        for j = 1:size(adders,1)
            G2(adders(j,1),adders(j,2)) = 1; G2(adders(j,2),adders(j,1)) = 1;
        end
        hold on
        p2 = plot(graph(G2)); p2.EdgeColor = "black"; p2.LineStyle = "--";
        p2.XData = xdat;  p2.YData = ydat;  p2.NodeLabel = {}; p2.MarkerSize = 0.1;
        G3 = zeros(size(G));
        for j = 1:size(breakers,1)
            G3(breakers(j,1),breakers(j,2)) = 1; G3(breakers(j,2),breakers(j,1)) = 1;
        end
        p3 = plot(graph(G3));
        p3.EdgeColor = "red"; p3.LineWidth = 5; p3.XData = xdat;
        p3.YData = ydat; p3.NodeLabel = {}; p3.MarkerSize = 0.1;
        f.Position = [100 100 300 300];
        saveas(f,path + sprintf("%s_pw_%d.eps",stab_text(us+1),su(us+1)),'epsc');
        close(); su(us+1) = su(us+1) + 1;
    end
end
```

Code A.12: Processes PW stability of given graphs and plots results. Uses A.10.

```
clear();
% script for the mc-pw comparison
J = 5;
sp.beta = 3*ones(J,1); sp.gamma = 2*ones(J,1);
sp.c = ones(J,1); sp.d = .2*ones(J,1); sp.P = 10;

gp.int_tol = 1e-8; gp.exhaustive_check = true;
gp.pure_bnb_check = false; gp.mod_bnb_check = false;
gp.fmincon_options = optimoptions(@fmincon,'Display', 'off','Algorithm','sqp');

a1 = load(sprintf("isomorphic_forest/J_%d.mat",J)).forest_list; % load forests
path = "stable_networks/n5_e1/";
process_mc_stability(a1, sp,  gp, J, path); process_pw_stability(a1,sp,J,path);


path = "stable_networks/n5_e2/";
sp.d = 1*ones(J,1);
process_mc_stability(a1, sp,  gp, J, path); process_pw_stability(a1,sp,J,path);
sp.d = .2*ones(J,1);


path = "stable_networks/n5_e3/";
sp.gamma = 10*ones(J,1);
process_mc_stability(a1, sp,  gp, J, path); process_pw_stability(a1,sp,J,path);

```

Code A.13: Script to compare PW and MC stability for 5 firms in Section 4.3. Uses A.11, A.12.

```
classdef recorder
    %RECORDER Holds turn records and yields relevant outputs from the records
    properties
        turn_records
        mcg_record
    end

    methods
        function obj = recorder(mcg_record)
            obj.turn_records = {};
            obj.mcg_record = mcg_record;
        end

        function  obj= add_turn_record(obj, record)
            obj.turn_records  =[obj.turn_records, {record}];
        end

        function obj = truncate(obj, I)
            if I>0
                obj.turn_records= obj.turn_records(1: end-I+1);
            end
        end

        function profits = get_profits(obj)
            profits = [];
            for i = 1:size(obj.turn_records,2)
                profits = [profits, obj.turn_records{i}.profits_n];
            end
        end

        function plot_profits(obj)
            profits = obj.get_profits();
            plot(profits')
        end
     end
end
```

Code A.14: A class that holds record objects of A.15.

```
classdef turn_record
    %TURN_RECORD Records information from a particular turn
    properties
        G_n; N_n; D_n; profits_n; i; pair_no;
    end
    methods
        function obj = turn_record(G, N, D,  profits,i)
            obj.G_n = G; obj.N_n = N; obj.D_n = D;
            obj.profits_n = profits; obj.i = i;
            obj.pair_no = ""; % for pairwise recording
        end
    end
end
```

Code A.15: Data class that records what happens in a turn of the formation procedure.

```
function [] = record_graphs(records, path)
%RECORD_GRAPHS Draws the records of the graph
%   turn_records: holds information from turns individually
%   path: file to be recorded
    profits = records.get_profits(); I = size(profits,1);
    x_tick_text = {}; x_tick_text_5 = {}; old_g = zeros(I);
    for i = 1:size(records.turn_records,2)
        this_g = records.turn_records{i}.G_n;
        if norm(this_g-old_g)>1e-3
            old_g = this_g; disp(sprintf("n = %d, is a difference",i-1))
        end
        this_i = records.turn_records{i}.i; f = figure;
        plot(graph(this_g),"EdgeColor","#101010","NodeColor","#101010")
        if records.mcg_record
            title(sprintf("n = %d, altered by firm %d",i-1,this_i))
        else
            title(sprintf("alteration %d by firms %s",i-1,records.turn_records{i}.pair_no))
        end
        f.Position = [100 100 200 200];
        saveas(f,path + sprintf("/net_%d.eps",i-1),'epsc');
        close(f);
        x_tick_text = [x_tick_text, {sprintf("%d",i-1)}];
        if mod(i,5)==1
            x_tick_text_5 = [x_tick_text_5, {sprintf("%d",i-1)}];
        else
            x_tick_text_5 = [x_tick_text_5, {""}];
        end
    end
    f = figure; imagesc(profits); colormap(1-[0:0.05:1;0:0.05:1;0:0.05:1]');
    caxis([min(min(profits))-1,max(max(profits))+1])
    if records.mcg_record
        title("firm profits over n"); xlabel("n");
    else
        title("firm profits after pairwise alterations"); xlabel("alterations");
    end
    ylabel("firms"); xticks(1:size(profits,2)); yticks(1:size(profits,1));
    colorbar; grid on;
    if size(profits,2)>8
        v = 600; xticklabels(x_tick_text_5);
    else
        v = 400; xticklabels(x_tick_text);
    end
    if I>10
        v2 = 400;
    else
        v2 = 200;
    end
    f.Position = [100 100 v v2];
    saveas(f,path + sprintf("/profits_visual.eps"),'epsc');
    close(f);
end
```

Code A.16: Function that plots records held in A.15.

```
clear();
sp.P = 10;

gp.int_tol = 1e-8;
gp.exhaustive_check = true;
gp.pure_bnb_check = false;
%gp.fmincon_options = optimoptions(@fmincon,'Display', 'off','Algorithm','interior-point');
gp.fmincon_options = optimoptions(@fmincon,'Display', 'off','Algorithm','sqp');
%gp.fmincon_options = optimoptions(@fmincon,'Display', 'off','Algorithm','active-set');


% 9 firms
graph_inputs_9_firms_eq;
graph_inputs_9_firms_var;

%15
graph_inputs_many_firms;


%for timing fmincon
if false
    I = 4;
    sp.beta = ones(I,1)*5;
    sp.gamma = ones(I,1)*2;
    sp.d = ones(I,1)*.2;
    sp.c = ones(I,1)*1;
    gp.exhaustive_check = false;
    records = gamer(sp, gp, false);
    gp.exhaustive_check = true;
end

%for timing minlp solvers
if false
    I = 19;
    sp.beta = ones(I,1)*5;
    sp.gamma = ones(I,1)*2;
    sp.d = ones(I,1)*.2;
    sp.c = ones(I,1)*1;
    gp.pure_bnb_check = true;
    records = gamer(sp, gp, false);
    gp.pure_bnb_check = false;
end
```

Code A.17: Main script used in Chapter 4. Uses, A.18, A.19, A.20.

```
address = "images/9_firms/";
sp.d = ones(9,1)*.2; sp.c = ones(9,1)*1; sp.beta = ones(9,1)*5; sp.gamma = ones(9,1)*2;

if false
    records = gamer(sp, gp, false); record_graphs(records, address+"equal_1");
end

if false
    rng(230001); records = gamer(sp, gp, true); record_graphs(records, address+"equal_2");
end

if false
    rng(230002); records = gamer(sp, gp, true); record_graphs(records, address+"equal_3");
end

if false
    rng(230003); records = gamer(sp, gp, true); record_graphs(records, address+"equal_4");
end

if false
    sp.beta = ones(9,1)*6.66; records = gamer(sp, gp, false);
    record_graphs(records, address+"equal_b"); sp.beta = ones(9,1)*5;
end


if false
    sp.beta = ones(9,1)*3; records = gamer(sp, gp, false);
    record_graphs(records, address+"equal_s"); sp.beta = ones(9,1)*5;
end

if false
    sp.gamma = ones(9,1)*4; records = gamer(sp, gp, false);
    record_graphs(records, address+"equal_q_b"); sp.gamma = ones(9,1)*2;
end

if false
    sp.gamma = ones(9,1)*1; records = gamer(sp, gp, false);
    record_graphs(records, address+"equal_q_s"); sp.gamma = ones(9,1)*2;
end

if false
    sp.c = ones(9,1)*3; records = gamer(sp, gp, false);
    record_graphs(records, address+"equal_c"); sp.c = ones(9,1)*1;
end

if false
    sp.d = ones(9,1)*1; records = gamer(sp, gp, false);
    record_graphs(records, address+"equal_d"); sp.d = ones(9,1)*.2;
end

if false
    sp.c = ones(9,1)*0; records = gamer(sp, gp, false);
    record_graphs(records, address+"equal_c_z"); sp.c = ones(9,1)*1;
end

if false
    sp.c = ones(9,1)*0;  sp.d = ones(9,1)*.5;  records = gamer(sp, gp, false);
    record_graphs(records, address+"equal_c_z2"); sp.c = ones(9,1)*1; sp.d = ones(9,1)*.2;
end

if false
    sp.d = ones(9,1)*0;  records = gamer(sp, gp, false);
    record_graphs(records, address+"equal_d_z"); sp.d = ones(9,1)*.2;
end

if false
    sp.c = ones(9,1)*3;  sp.d = ones(9,1)*0; records = gamer(sp, gp, false);
    record_graphs(records, address+"equal_d_z2"); sp.c = ones(9,1)*1;  sp.d = ones(9,1)*.2;
end
```

Code A.18: Script from A.17 for 9 firm simulations part 1.

```
address = "images/9_firms/";
sp.d = ones(9,1)*.2;
sp.c = ones(9,1)*1;
sp.gamma = ones(9,1)*2;

if false
    sp.beta = linspace(2,6,9)';
    records = gamer(sp, gp, false);
    record_graphs(records, address+"var_b_1");
end

if false
    rng(230011)
    sp.beta = linspace(2,6,9)';
    records = gamer(sp, gp, true);
    record_graphs(records, address+"var_b_2");
end

if false
    rng(230012)
    sp.beta = linspace(2,6,9)';
    records = gamer(sp, gp, true);
    record_graphs(records, address+"var_b_3");
end

if false
    rng(230013)
    sp.beta = linspace(2,6,9)';
    records = gamer(sp, gp, true);
    record_graphs(records, address+"var_b_4");
end

if false
    sp.beta = [2;2;2;2;4;6;6;6;6];
    records = gamer(sp, gp, false);
    record_graphs(records, address+"var_bd_1");
end

if false
    rng(230021)
    sp.beta = [2;2;2;2;4;6;6;6;6];
    records = gamer(sp, gp, true);
    record_graphs(records, address+"var_bd_2");
end

if false
    rng(230022)
    sp.beta = [2;2;2;2;4;6;6;6;6];
    records = gamer(sp, gp, true);
    record_graphs(records, address+"var_bd_3");
end

if false
    rng(230023)
    sp.beta = [2;2;2;2;4;6;6;6;6];
    records = gamer(sp, gp, true);
    record_graphs(records, address+"var_bd_4");
end
```

Code A.19: Script from A.17 for 9 firm simulations part 2.

```
address = "images/many_firms/";
sp.beta = ones(16,1)*5;
sp.gamma = ones(16,1)*2;

if false
    sp.d = ones(16,1)*.2;
    sp.c = ones(16,1)*1;
    records = gamer(sp, gp, false);
    record_graphs(records, address+"equal_1");
end

if false
    rng(230101)
    sp.d = ones(16,1)*.2;
    sp.c = ones(16,1)*1;
    records = gamer(sp, gp, true);
    record_graphs(records, address+"equal_2");
end

if false
    rng(230121)
    sp.d = [ones(4,1)*.1;ones(4,1)*.1;ones(4,1)*.5;ones(4,1)*.5];
    sp.c = [ones(4,1)*.5;ones(4,1)*2;ones(4,1)*.5;ones(4,1)*2];
    records = gamer(sp, gp, true);
    record_graphs(records, address+"var_1");
end

if false
    rng(230122)
    sp.d = [ones(4,1)*.1;ones(4,1)*.1;ones(4,1)*.5;ones(4,1)*.5];
    sp.c = [ones(4,1)*.5;ones(4,1)*2;ones(4,1)*.5;ones(4,1)*2];
    records = gamer(sp, gp, true);
    record_graphs(records, address+"var_2");
end

if false
    rng(230123)
    sp.d = [ones(4,1)*.1;ones(4,1)*.1;ones(4,1)*.5;ones(4,1)*.5];
    sp.c = [ones(4,1)*.5;ones(4,1)*2;ones(4,1)*.5;ones(4,1)*2];
    records = gamer(sp, gp, true);
    record_graphs(records, address+"var_3");
end
```

Code A.20: Script from A.17 for simulations with a larger number of firms.