university of groningen

faculty of science and engineering

Master Thesis

System Identification in Network Games

by

Bastiaan Kok

Submitted in Partial Fulfilment of the Requirements for the Degree
MSc Industrial Engineering and Management

Faculty of Science and Engineering

Supervisors:
dr. N. Monshizadeh
dr. A.K. Cherukuri
M. Shakarami (PhD)

Groningen
March 29, 2022

Abstract

In this research, we consider the problem of identifying network parameters that define a network game from data. The players in a network game interact continuously with each other, creating an action profile over time. These actions are captured by a central regulator who performs system identification to estimate the network parameters. In this report an algorithm is provided to perform the system identification using an adaptive lasso that promotes sparsity in the solution. This is a desired property as the graph that defines the interaction network of a network game is expected to be sparse, as links indicate how players are influenced by each other. Using the identified network parameters, the central regulator can apply suitable interventions to steer the network to social welfare. Results from experiments showed that the adaptive lasso is a robust algorithm that can identify network parameters in an underdetermined setting, and when there is noise present in the network. A drawback is that the adaptive lasso is computationally expensive to optimise when the number of players in a network is large, or the sample size is large.

Contents

1 Introduction **1**

1.1 Problem Analysis 2

1.2 Research Objective 2

1.3 Research Questions 3

1.4 Research Outline 3

2 Theoretical Background **4**

2.1 Network Game Formulation 4

2.1.1 Linear quadratic games 4

2.1.2 Player's action dynamics 5

2.1.3 Equilibria 5

2.1.4 Intervention protocol 6

3 System Identification **7**

3.1 Data Gathering 7

3.1.1 Euler Discretization 7

3.1.2 Data Informativity 7

3.1.3 Noise 8

3.2 Candidate Model 8

3.3 Singular Value Decomposition 9

3.4 Linear Regression 10

3.5 Regularised Linear Regression 11

3.5.1 Problems with Lasso 12

3.6 Adaptive Lasso 13

3.7 Cross-validation 13

4 Simulation Experiments **15**

4.1 Model Set-up 15

4.2 Experimental Set-up 16

4.2.1 Varying sample size 16

4.2.2 Adaptive lasso in under- and overdetermined systems 17

4.2.3 Link failure detection 18

5 Simulation Results **21**

5.1 Results varying sample size 21

5.2 Results adaptive in under- and over-determined systems 24

5.3 Results link failure 26

6 Conclusion **28**

7 Discussion **29**

7.1 Future Work 30

Appendices **33**

A Data Matrix Sample Size Variation **33**

Chapter 1

Introduction

Network games provide a powerful framework for formal analysis of a wide variety of social and economic settings. In general, individuals decisions are not merely based on their own judgement, but are rather influenced by the actions of their peers. For example, the decision of an individual to buy a new product, commit a crime, or contribute to a public good is affected by the choice of people in their social network (Parise and Ozdaglar, 2021). Hence, the pay-off of individuals i.e. players/agents is based on their own actions and the actions of their neighbours, creating a standalone marginal return. We can study these games by defining the underlying interaction network using graphs, where the pairwise links can represent geographic distance or peer effects (Bramoullé and Kranton, 2015). Engineering applications include supply chain network game modelling for competing firms, from product differentiation and price competition (Nagurney and Li, 2015) to labour constraints (Nagurney, 2021).

The players in the network are considered independent, non-cooperative and rational, meaning that they show selfish behaviour in maximising their own pay-off, evoking a game theoretic approach in their analysis. Therefore, players will eventually converge to a Nash equilibrium, a situation in which each player is taking his optimal action with respect to the actions taken by all other players. No player has an incentive to deviate from his current action, as it will decrease his pay-off (Koutsoupias and Papadimitriou, 1999).

Such selfish behaviour of players leads to deterioration of performance with respect to the situation where all players would cooperate to maximise social welfare, which is defined as the sum of the pay-off of all players (Shakarami et al., 2021). This degradation of performance, or loss in efficiency, is defined by the price of anarchy (Koutsoupias and Papadimitriou, 1999). Deviating from the Nash equilibrium to the social optimum is achieved through interventions. A central regulator who is aware of the underlying interaction network can provide incentive for the players to deviate from their current action by modifying their standalone marginal return. Subsequently, the players choose different strategies and are steered towards the social optimum (Shakarami et al., 2021).

The problem that arises is that the central regulator must have access to the agents' private information to apply suitable interventions. Generally speaking, there are many situations in which this network information is unknown to the agents or the central regulator (Galeotti et al., 2010). For instance due to privacy or proprietary concerns (Parise and Ozdaglar, 2021) Therefore, before a central regulator can apply suitable interventions, it is necessary to determine the underlying interaction network and the standalone marginal return. There do exist control techniques to steer the agent towards the social optimum without perfect information, however, these techniques require at least knowledge of the underlying interaction network, or the social optimum (Alpcan et al., 2009). Furthermore, Shakarami et al. (2021) proposed a dynamic intervention protocol, that requires the player's standalone marginal return alone. During this research, these network parameters are considered unknown to the central regulator. Resulting in the need to develop methods to recover these parameters before interventions can be applied to the network.

System identification is a process of going from observed data to a mathematical model. It has its roots in statics and was early adopted by the control engineering community. The objective is finding dynamical models based on differential equations from observed data. The system identification procedure is characterised by four requirements. Observed data, a candidate model, a criterion of fit and validation (Ljung, 1998). Studying a system identification technique to retrieve the network parameters from observed data is the main topic of this research.

Therefore, in this research we present a system identification technique that will retrieve the network parameters for quadratic network games, based on the observed actions of players interacting in a network. The data is obtained from a network game which is used as an illustrative example of a six player network game in (Shakarami et al., 2021). In this example, the agents are selfish and interested in maximising their individual pay-off function. They choose their actions based on pseudo-gradient dynamics. The performance of the identification algorithm is assessed by using the identified parameters as input for the central regulator and determine if the network is steered towards the social optimum.

1.1 Problem Analysis

Many social or economic settings can be modelled using network games. In general, the network parameters that define the network are unknown to a central regulator. Through the price of anarchy, the action profile of network games tends to converge to a sub-optimal Nash equilibrium, where no player has a unilateral profitable deviation. However, there exists an action profile that increases the total pay-off of all agents leading to social welfare. Social welfare can only be achieved through interventions from a central regulator. Subsequently, these interventions can only be applied if the central regulator has full knowledge of the network parameters.

1.2 Research Objective

From a utilitarian perspective it is desired to steer the action profile of quadratic network games to the solution of the social welfare maximisation problem. To achieve this objective, it is required to have knowledge of the network parameters to apply a suitable intervention. Therefore, the objective of this research is to identify the network parameters of a network game from observed data. The system identification technique is tested on clean data, as well as data corrupted by noise.

1.3 Research Questions

The main research question that stands central throughout this research is the following: "*How are the network parameters of quadratic network games determined from observed data under noisy conditions?*". To find the solution to this research questions, the following sub questions are derived.

1. What are linear quadratic network games and how are they defined mathematically?
2. How is suitable data obtained from simulating network games?
3. What is a suitable system identification technique to derive a mathematical model from observed data?
4. What is the effect of noisy data and how do we deal with noise?

1.4 Research Outline

The thesis is organised using seven chapters. Chapter 2 provides a theoretical background on how to formulate network games mathematically. Chapter 3 discusses system identification techniques and properties of the required data. Chapter 4 describes the simulation experiments used to assess the performance of the proposed algorithms. Chapter 5 presents the simulation results, and chapter 6 and 7 provide concluding remarks and a discussion and future work.

Notation:
The symbol $\mathbb{R}^n$ denotes a set of real numbers of dimension n. The norm of a vector is denoted by $\|\cdot\|_p$, where $p \in \{0, 1, 2\}$. Vectors and matrices are presented in bold face, and matrices are denoted by a capital letter. For a given vector $\mathbf{x} \in \mathbb{R}^n$ its i^{th} element is denoted by x_i. The transpose of a vector $\mathbf{y}$ is denoted by $\mathbf{y}^\top$. The words player/agent are used interchangeably throughout this thesis.

Chapter 2

Theoretical Background

2.1 Network Game Formulation

We consider a game with a finite set of players $\mathcal{I} = \{1, ..., n\}$ that interact with each other continuously on an underlying interaction network. The connections between the players are defined by the adjacency matrix $\mathbf{P} \in \mathbb{R}^{n \times n}$, where $P_{ij} \in [0, 1]$ denotes the influence of player j on player i. The network contains no self-loops, therefore $P_{ii} = 0 \quad \forall i \in \mathcal{I}$. The set of neighbours of player i is denoted by $\mathcal{N}_i = \{j \in \mathcal{I} \mid P_{ij} > 0\}$. The network is undirected if $P_{ij} = P_{ji}$ for all i, j, otherwise it is directed.

The objective of each player is to maximise his payoff function denoted by $U_i(x_i, z_i(x), u_i)$, which captures his own action $x_i \in \mathbb{R}$ and the aggregate of his neighbours actions (Shakarami et al., 2021)(Parise and Ozdaglar, 2021). The network aggregate of player i is defined by:

$$z_i(x) := \sum_{j=1}^{n} P_{ij} x_j. \tag{2.1}$$

2.1.1 Linear quadratic games

Linear quadratic games are used in literature to model peer effects in social and economic settings (Engwerda, 2006). In linear quadratic games, each player has a payoff that is quadratic in his own strategy and linear in the network aggregate (Parise and Ozdaglar, 2021). These so-called linear quadratic games are popular in the literature for practical considerations. On the one hand, these type of games are analytically and numerically solvable, which becomes much more difficult when considering nonlinear partial differential equations. On the other hand, linear quadratic problem settings appear naturally in situations where the player is trying to minimise the effect of small perturbations in their environment (Engwerda, 2005). A parametric formulation of the utility function is given in Equation 2.2 and allows for a tractable analysis of the impact of the network structure and equilibrium analysis (Parise and Ozdaglar, 2021).

$$U_i(x_i, z_i(x), u_i) := -\frac{1}{2}x_i^2 + x_i(az_i(x) + b_i) + x_i u_i, \tag{2.2}$$

where $a \in \mathbb{R}$ captures the impact of the network aggregate and $b_i \in \mathbb{R}$ is the standalone marginal return. Furthermore, the term $x_i u_i$ is used to capture the effect of the central regulator which modifies the standalone marginal return b_i into $b_i + u_i$.

An interesting property of the pay-off parameter a is that its sign (positive or negative) defines whether the game exhibits increasing (or decreasing) pay-off differences. These are so-called games of strategic complements and strategic substitutes (Galeotti et al., 2010). For instance, strategic complements arise whenever a player obtains a greater pay-off when his neighbours play the same strategy. Strategic substitutes arise whenever the opposite is taking place such as when one player is providing more to a public good, his neighbour starts providing less, which is known as free-riding.

2.1.2 Player's action dynamics

All players are rational in choosing their action and interested in maximising their individual pay-off function U_i given the aggregated behaviour of their neighbours $z_i(x)$, and the intervention signal u_i. Therefore, the players continuously take action until they reach an equilibrium. De Persis and Grammatico (2019) introduced the pseudo-gradient dynamics that describe the evolution of the players' actions over time by taking the derivative of the players' pay-off function, which results in the following dynamics:

$$\dot{x}_i(t) = \frac{\partial U_i}{\partial x_i}\left(x_i(t), z_i(x(t)), u_i(t)\right), \quad \forall\, i \in \mathcal{I}. \tag{2.3}$$

By computing this derivative we obtain the action dynamics for every player i.

$$\begin{aligned} \dot{x}_i(t) &= \frac{\partial U_i}{\partial x_i}\left(-\frac{1}{2}x_i^2 + x_i(az_i(x) + b_i) + x_i u_i\right), && \text{(2.4a)}\\ &= -x_i(t) + az_i(x(t)) + b_i + u_i(t), && \text{(2.4b)}\\ &= -x_i(t) + a\sum_{j=1}^{n} P_{ij}x_j(t) + b_i + u_i(t). && \text{(2.4c)} \end{aligned}$$

The pseudo gradient dynamics in Equation 2.4c can be written in matrix form to capture the complete network dynamics.

$$\dot{\mathbf{x}}(t) = (-\mathbf{I} + a\mathbf{P})\mathbf{x}(t) + \mathbf{b} + \mathbf{u}(t). \tag{2.5}$$

2.1.3 Equilibria

For this type of network games, there exist two types of equilibria of interest. One possible equilibrium is when $u(\cdot) = 0$, there is no intervention applied by the central regulator. In that case, the network will converge to the Nash equilibrium of the game, where each agent plays the best response to other agents actions. In this situation no player will deviate from its best response as it will result in a lower pay-off (Parise and Ozdaglar, 2021). However, the Nash equilibrium is not necessarily the optimal situation where all players achieve their maximum pay-off possible. Therefore, we introduce the social optimum which describes a state where all agents still play their best response to other agents actions, but obtain a higher total pay-off. To reach the social optimum, the central regulator must apply suitable interventions that steer the agents to the social optimum and reach social welfare. The social optimum is defined by the following social welfare maximisation problem:

$$x_{opt} := \underset{x\in\mathbb{R}^n}{\operatorname{argmax}} \sum_{i=1}^{n} -\frac{1}{2}x_i^2 + x_i(az_i(x) + b_i). \tag{2.6}$$

To explore the existence of a unique social optimum x_{opt}, Shakarami et al. (2021) proposed a necessary and sufficient condition which must be met, namely:

$$\max_{i \in \mathcal{I}} a\lambda_i(\mathbf{P} + \mathbf{P}^\top) < 1, \tag{2.7}$$

where λ_i are the eigenvalues of the matrix $\mathbf{P} + \mathbf{P}^\top$. When this condition is satisfied, the unique maximiser is given by:

$$\mathbf{x}_{opt} = \Big(\mathbf{I} - a(\mathbf{P} + \mathbf{P}^\top)\Big)^{-1}\mathbf{b}. \tag{2.8}$$

2.1.4 Intervention protocol

The objective of the central regulator is to steer the network to the social optimum by applying interventions that modify the players standalone marginal return. The intervention protocol used throughout this is static feedback intervention. Static feedback can be applied when the central regulator has complete knowledge of the network and pay-off parameter $a\mathbf{P}$. It ensures convergence to the social optimum and is denoted by $\mathbf{u}(t) = a\mathbf{P}^\top\mathbf{x}(t)$. Under static feeback intervention, the action profile of the agents converges exponentially to the social optimum $\mathbf{x}_{opt}$, which is proven by Shakarami et al. (2021).

Chapter 3

System Identification

The previous chapter provided a detailed mathematical formulation on network games. In this chapter it is described how suitable data for system identification is obtained. Then, two methods are proposed that can perform the system identification.

3.1 Data Gathering

Before a system identification method is proposed, we describe how the relevant data will be gathered. Then, to increase the informativity of the data, we define an excitation signal that is sufficiently rich. Finally, we describe how the data can be corrupted by noise, which complicates the system identification process.

3.1.1 Euler Discretization

The pseudo-gradient dynamics in Equation 2.5 are given in continuous time. However, to simulate these dynamics, discrete time dynamics are required. Therefore, Euler's method is used for discretisation of the dynamics. Euler's method is a numerical solution to a differential equation and its derivation is given in Equation 3.1.

$$x\left(t_{k+1}\right)=x\left(t_k\right)+\int_{t_k}^{t_{k+1}} f(t, x(t)) \mathrm{d} t \approx x_k+\int_{t_k}^{t_{k+1}} f\left(t_k, x_k\right) \mathrm{d} t=x_k+h f\left(t_k, x_k\right)=x_{k+1}. \tag{3.1}$$

The computed values of $\mathbf{x}_{k+1}$ are an approximation of the true values $\mathbf{x}(t_{k+1})$. Choosing a smaller step size h requires more computations, but leads to a more accurate approximation of the ODE.

Applying Euler's discretization method to the pseudo-gradient dynamics in Equation 2.5 results in Equation 3.2. These discretised dynamics are used to simulate network games, and analyse output data in the form of information on the player's strategy over time.

$$\mathbf{x}_{k+1}=\mathbf{x}_k+h((-\mathbf{I}+a\mathbf{P})\mathbf{x}_k+\mathbf{b}+\mathbf{u}). \tag{3.2}$$

3.1.2 Data Informativity

To obtain a consistent estimate of the prediction parameters in system identification, data is required to be informative with respect to the model structure (Gevers et al., 2008). In the open loop system in Equation 3.2, data informativity is equivalent to the persistency of the input regressor (i.e. the regressor is persistently exciting) derived from the model structure. A necessary and sufficient condition for the input regressor to be persistently exciting is that the the signals in the regressor are linearly independent (Colin et al., 2019).

For this application we consider the case where the input regressors $u_i(t)$ are multisine. Colin et al. (2019) deduced that one necessary condition on $u_i(t)$ is that each sinusoid has two frequencies in its power spectral density, hence, each $u_i(t)$ must contain at least $m/2$ sinusoids at different frequencies. Here, m is the order of the model structure. Take $n \geq m/2$ sinusoids at different frequencies denoted by $[\omega_1 \dots \omega_h]$, where $h \in [1, n]$. Then, each regressor input $u_i(t)$ is given by the following formulation.

$$u_i(t) = \sum_{h=1}^{n} a_{ih} \sin(\omega_h t + \phi_{ih}), \tag{3.3}$$

where, a_{ih} are the amplitudes, ω_h are the frequencies and ϕ_{ih} are phase shifts.

3.1.3 Noise

The challenge that arises when performing system identification in the case of noise is that exact recovery of the underlying system is impossible. For this thesis we consider endogenous noise, which is noise present within the network and can be thought of as players that obtain a noisy measurement of other players actions. If players have noisy measurements on other players' actions, their optimal response is different than their optimal response when reacting to the true action of other players. Therefore, endogenous noise is affecting the action profile of the players, resulting in more complex network dynamics. We model the noise as a uniform distributed variable with zero mean ϵ_k. The noise term is added to the differential equation resulting in:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + h((-\mathbf{I} + a\mathbf{P})\mathbf{x}_k + \mathbf{b} + \mathbf{u} + \boldsymbol{\epsilon}_k). \tag{3.4}$$

A uniform distributed random variable has an arbitrary outcome that lies between two bounds. The bounds are given by a, b and determine the minimum and maximum possible outcome. The probability that a variable takes a certain value is equal for all values within this interval. The probability density function of a uniform distribution is given by Equation 3.5.

$$f(x) = \begin{cases} \frac{1}{b-a} & \text{for } a \leq x \leq b \\ 0 & \text{otherwise.} \end{cases} \tag{3.5}$$

For simulation purposes, the bounds of the uniform distribution are defined. However, as the number of random variables used for the experiments is finite, a set of random variables is defined by its mean and variance. In terms of mean (μ) and variance (σ^2) the probability density function is given by Equation 3.6.

$$f(x) = \begin{cases} \frac{1}{2\sigma\sqrt{3}} & \text{for } -\frac{1}{\sigma\sqrt{3}} \leq x - \mu \leq \frac{1}{\sigma\sqrt{3}} \\ 0 & \text{otherwise.} \end{cases} \tag{3.6}$$

3.2 Candidate Model

To identify the network parameters using system identification, a simple candidate mathematical model of linear equations that fits the data is required, in order to perform linear regression.

Therefore, we rewrite the discretized pseudo-gradient dynamics in Equation 3.2 to the simple linear form in Equation 3.7d.

$$\begin{aligned}
\mathbf{x}_{k+1} &= \mathbf{x}_k + h((-\mathbf{I} + a\mathbf{P})\mathbf{x}_k + \mathbf{b} + \mathbf{u}) && (3.7a)\\
&= \mathbf{x}_k + h(\mathbf{A}x_k + \mathbf{b} + \mathbf{u}) && (3.7b)\\
&= (\mathbf{I} + h\mathbf{A})\mathbf{x}_k + h\mathbf{b} + h\mathbf{u} && (3.7c)\\
&= \tilde{\mathbf{A}}\mathbf{x}_k + \tilde{\mathbf{b}} + \tilde{\mathbf{u}}, && (3.7d)
\end{aligned}$$

where $\mathbf{A} = (-\mathbf{I} + a\mathbf{P})$, $\tilde{\mathbf{A}} = (\mathbf{I} + h\mathbf{A})$, $\tilde{\mathbf{b}} = h\mathbf{b}$ and $\tilde{\mathbf{u}} = h\mathbf{u}$.

In this current form it remains difficult to perform linear regression if we have a large data set containing the state vectors for a certain time period. Therefore, it is necessary to write Equation 3.7d in a single map such that it becomes easier to fit the model to the data. Consider $\mathbf{u} = \mathbf{0}$ for now.

$$\mathbf{x}_{k+1} = \tilde{\mathbf{A}}\mathbf{x}_k + \tilde{\mathbf{b}} = \begin{bmatrix} \tilde{A}_{11} & \dots & \tilde{A}_{1n} \\ \vdots & \ddots & \vdots \\ \tilde{A}_{n1} & \dots & \tilde{A}_{nn} \end{bmatrix} \begin{bmatrix} x_{1k} \\ \vdots \\ x_{nk} \end{bmatrix} + \begin{bmatrix} \tilde{b}_1 \\ \vdots \\ \tilde{b}_n \end{bmatrix} \tag{3.8}$$

$$\Rightarrow \mathbf{x}_{k+1} = \begin{bmatrix} \tilde{A}_{11} & \dots & \tilde{A}_{1n} & \tilde{b}_1 \\ \vdots & \ddots & \vdots & \vdots \\ \tilde{A}_{n1} & \dots & \tilde{A}_{nn} & \tilde{b}_n \end{bmatrix} \begin{bmatrix} x_{1,k} \\ \vdots \\ x_{n,k} \\ 1 \end{bmatrix} \tag{3.9}$$

$$\Rightarrow \mathbf{x}_{k+1} = \hat{\tilde{\mathbf{A}}} \begin{bmatrix} \mathbf{x}_k \\ 1 \end{bmatrix}. \tag{3.10}$$

3.3 Singular Value Decomposition

The singular value decomposition (SVD) is among the most useful matrix factorisations providing a foundation for many statistical methods. The SVD provides a numerically stable matrix that is guaranteed to exist and has two important characteristics that are used throughout this research. The SVD can be used to perform a Moore-Penrose pseudo-inverse of a non-square matrix which is used to find the least squares solution to a system of equations $\mathbf{A}\mathbf{x} = \mathbf{b}$. Another useful feature of the SVD is that it provides an optimal low-rank approximations to a matrix by keeping the leading singular values and vectors, and discarding the rest (Brunton and Kutz, 2019).

Consider a real non-square matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ with rank r, then $\mathbf{A}$ can be rewritten as a product of three matrices: $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$, where $\mathbf{U}$ is an $m \times m$ orthogonal matrix and $\mathbf{V}$ is an $n \times n$ orthogonal matrix, i.e. $\mathbf{U}^\top\mathbf{U} = \mathbf{I}$ and $\mathbf{V}^\top\mathbf{V} = \mathbf{I}$. $\mathbf{\Sigma}$ is an $m \times n$ pseudo diagonal matrix.

The first r elements on the diagonal of $\mathbf{\Sigma}$ are the singular values of $\mathbf{A}$. All other elements of $\mathbf{\Sigma}$ are zero. The matrix $\mathbf{A}^\top\mathbf{A}$ is a symmetric matrix that can be written as $\mathbf{A}^\top\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$, where $\mathbf{V}$ represents the right singular values of $\mathbf{A}$ and $\mathbf{\Lambda}$ is a diagonal matrix consisting of the eigenvalues of $\mathbf{A}\mathbf{A}^\top$. The singular values are the positive square roots of the eigenvalues of $\mathbf{A}^\top\mathbf{A}$ i.e. $\sigma_i = \sqrt{\lambda_i}$ for $i = 1, 2, ..., r$. The SVD of a matrix can be used to compute a pseudo-inverse of that matrix according to $\mathbf{A}^\dagger := \mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^\top$.

The system $\mathbf{Ax} = \mathbf{b}$ is overdetermined when $\mathbf{A} \in \mathbb{R}^{m\times n}$ with $m > n$, meaning that there are more equations than unknowns. This matrix cannot be full column rank because the columns of the matrix are not linearly independent. Hence, it is guaranteed that there are vectors $\mathbf{b}$ that have no solution $\mathbf{x}$. However, in this overdetermined case, it is often desired to find a solution $\mathbf{x}$ that minimises the sum of the squared error $||\mathbf{Ax} - \mathbf{b}||_2$, called the *least-squares* solution (Brunton and Kutz, 2019). The solution to this optimisation problem is found by computing $\mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{U}^\top\mathbf{b}$.

The SVD is a great choice for solving this optimisation problem as computing the pseudo-inverse is computationally efficient, after computing the more expensive SVD. Inverting the matrices $\mathbf{U}, \mathbf{V}$ involves matrix multiplications requiring $\mathcal{O}(n^2)$ operations (Brunton and Kutz, 2019).

3.4 Linear Regression

Using Equation 3.10 it is now possible to solve for $\hat{\tilde{\mathbf{A}}}$ by applying a right singular value decomposition. The required data matrices $\mathbf{X}$ and $\mathbf{X}'$ are constructed as follows:

$$\begin{aligned} \mathbf{X} &= \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_{m-1} \\ 1 & 1 & \dots & 1 \end{bmatrix} \\ \mathbf{X}' &= \begin{bmatrix} \mathbf{x}_2 & \mathbf{x}_3 & \cdots & \mathbf{x}_m \end{bmatrix}, \end{aligned} \tag{3.11}$$

where data matrix $\mathbf{X}$ contains the snapshots of the state vectors and data matrix $\mathbf{X}'$ contains snapshots of the time shifted state vectors. Writing Equation 3.13 for the entire data set, and rewriting such that we solve for $\hat{\tilde{\mathbf{A}}}$ results in the following:

$$\begin{aligned} \mathbf{X}' &= \hat{\tilde{\mathbf{A}}}\mathbf{X} \\ \Rightarrow \hat{\tilde{\mathbf{A}}} &= \mathbf{X}'\mathbf{X}^\dagger, \end{aligned} \tag{3.12}$$

where $\mathbf{X}^\dagger$ is the pseudo inverse and $\hat{\tilde{\mathbf{A}}}$ is denoted by:

$$\hat{\tilde{\mathbf{A}}} = \begin{bmatrix} \tilde{\mathbf{A}} & \tilde{\mathbf{b}} \end{bmatrix}. \tag{3.13}$$

Note, that when an exploration signal is used in the simulation of the network game, this must be accounted for in the linear regression. Therefore:

$$\mathbf{X}' - h\mathbf{U}_{expl} = \hat{\tilde{\mathbf{A}}}\mathbf{X} \tag{3.14}$$

where $\mathbf{U}_{expl}$ denotes a matrix containing the exploration signal $\mathbf{U}_{expl} = [\mathbf{u}_1\mathbf{u}_2 \cdots \mathbf{u}_{m-1}]$.

Solving the system of linear equations in Equation 3.12 using an SVD allows us to retrieve matrix $\tilde{\mathbf{A}}$ and the vector $\tilde{\mathbf{b}}$. $\tilde{\mathbf{A}}$ contains the network structure as $\mathbf{I} + h(-\mathbf{I} + a\mathbf{P})$, and $\tilde{\mathbf{b}}$ contains the marginal return vector as $h\mathbf{b}$.

Now by back substituting for $\tilde{\mathbf{A}}$, we obtain $\tilde{\mathbf{A}} = \mathbf{I} + h\mathbf{A} = \mathbf{I} + h(-\mathbf{I} + a\mathbf{P})$. To uncover the network information $a\mathbf{P}$, we invert the equation resulting in $a\mathbf{P} = ((\tilde{\mathbf{A}} - \mathbf{I})/h) + \mathbf{I}$. To uncover the marginal return $\mathbf{b}$, we simply divide $\tilde{\mathbf{b}}$ by the time step h.

3.5 Regularised Linear Regression

Exact system identification using the the pseudo inverse only works in the ideal situation when there is no noise present in the network ($\epsilon = \mathbf{0}$). When there is noise present in the network ($\epsilon \neq \mathbf{0}$), the method does not result in the true solution due to the pseudo inverse, which is a method of finding the best-fit or least squares solution (Brunton and Kutz, 2019). This means that the found solution will be non-sparse, meaning that all predictor variables are nonzero. However, for this application we require the solution to be sparse as the network defined using a graph is sparse.

Consider the following linear regression model:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta}^* + \boldsymbol{\epsilon}, \tag{3.15}$$

where $\mathbf{y}$ is an $n \times 1$ response vector, $\mathbf{X}$ is an $n \times p$ covariate matrix, $\boldsymbol{\beta}^*$ is the regression vector and $\boldsymbol{\epsilon}$ is an $n \times 1$ error matrix of independent identically distributed (i.i.d.) random variables with zero mean and variance σ^2.

Finding the sparsest solution to a linear regression problem as defined in Equation 3.15 is based on an l_0 quasi-norm $\|\cdot\|_0$ which counts the number of nonzero elements in its argument, and can be formulated as a mathematical program (Tropp, 2006).

$$\begin{aligned} \underset{\boldsymbol{\beta}}{\text{minimise}} \quad & \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2 \\ \text{subject to} \quad & \|\boldsymbol{\beta}\|_0 \leq \xi. \end{aligned} \tag{3.16}$$

Solving Equation 3.16 requires combinatorial optimisation which is computationally intractable due to the exponentially large search space. To mitigate this problem, we replace the l_0 quasi-norm with an l_1 norm to obtain a convex optimisation problem (Tropp, 2006).

$$\begin{aligned} \underset{\boldsymbol{\beta}}{\text{minimise}} \quad & \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2 \\ \text{subject to} \quad & \|\boldsymbol{\beta}\|_1 \leq \xi. \end{aligned} \tag{3.17}$$

This substitution is referred to as convex relaxation, as the l_1 norm is a convex function closest to the l_0 quasi-norm. The advantage of the new formulation is that it is computationally tractable and can be solved in polynomial time. Furthermore, when the solution to the relaxation is sparse, it can be a good approximation of the ideal coefficient vector $\boldsymbol{\beta}^*$ (Donoho, 2006). This technique is called least absolute shrinkage and selection operator, or lasso for short, and is used to impose a size constraint on the coefficients to promote sparsity in the solution Van Le (2020).

In Equation 3.17 ξ is a predetermined free variable that determines the amount of regularisation. The lasso is written in a more compact in Lagrangian form in Equation 3.18. Lagrangian duality implies that there is a one-to-one correspondence between the constrained optimisation Equation 3.17 and the Lagrangian form Equation 3.18. More precisely, there is a corresponding value of λ for each value of ϵ such that both Equation 3.17 and Equation 3.18 yield the same solution (Van Le, 2020).

$$\hat{\boldsymbol{\beta}}^{*} = \arg\min_{\boldsymbol{\beta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2 + \lambda \|\boldsymbol{\beta}\|_1 . \tag{3.18}$$

The parameter λ is varied through a range of values to obtain the best fit. The fit is validated against a test set using k-fold cross validation. Because the constraint region is diamond shaped, choosing a sufficiently large value of λ is likely to result in a solution that lies at the corner point of that region. As a result, a sparse solution is obtained in which some coefficients are set to zero (Van Le, 2020). This is illustrated in Figure 3.1. This example is only valid in 2D; the constraint region is a polyhedron in 3D and a polytope in higher dimensions, however, the key insights in Figure 3.1 still hold (James et al., 2013).

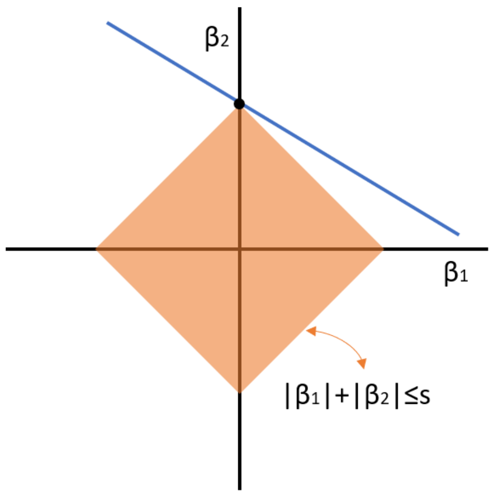


Figure 3.1: Due to the diamond shaped constraint region in 2D, choosing λ sufficiently large results in a sparse solution where one or more coefficients are zero.

3.5.1 Problems with Lasso

Experimenting with the lasso for system identification was not giving satisfactory results. The main problem concerning lasso was that it failed to select the proper model as predictor variables are shrunken too much. False discoveries occur early along the lasso path, therefore, lasso is considered merely as a variable screener, rather than a model selector (Tibshirani, 1996).

These problems with lasso are usually attributed to small effect sizes or correlations between the predictor variables. However, Su et al. (2017) showed that lasso fails to select the proper model, even when the signal to noise ratio is infinitely large and the predictor variables are stochastically independent. Therefore, lasso is regarded as unsuitable for identifying the correct model for system identification.

3.6 Adaptive Lasso

Whereas a regular lasso algorithm fails to identify the correct model, there exists a slight variation to this algorithm called adaptive lasso. In the regular lasso algorithm each regressor coefficient is penalised by the same regularisation parameter λ in the l_1 penalty, thus shrinking each coefficient equally. By applying different weights to each regressor variable, this problem is mitigated. This method of penalising each regressor coefficient differently is called the adaptive lasso and was first described by (Zou, 2006). When the weights of the adaptive lasso are data driven and cleverly chosen, this method is suitable for model selection.

Consider the linear regression model in Equation 3.15, then take an estimator $\hat{\boldsymbol{\beta}}$ that is a root-n-consistent estimator of $\boldsymbol{\beta}^*$, for instance pick $\hat{\boldsymbol{\beta}}$ that gives the ordinary least squares solution to the regression problem. Pick $\gamma > 0$, and define the weight vector according to $\mathbf{w} = 1/|\hat{\boldsymbol{\beta}}|^\gamma$. Then, the adaptive lasso is given by:

$$\hat{\boldsymbol{\beta}}^* = \arg\min_{\beta} \left\| \mathbf{y} - \sum_{j=1}^{p} \mathbf{x}_j \beta_j \right\|_2 + \lambda_n \sum_{j=1}^{p} \hat{w}_j \left\| \beta_j \right\|_1 . \tag{3.19}$$

The adaptive lasso provides improvement over regular lasso in that it does not require strong regularity conditions on the regressor variables for variable selection, and reduces the bias in the selected nonzero components (Chatterjee and Lahiri, 2013). To show these properties, suppose that the first p_0 elements of the true regression variable $\boldsymbol{\beta}^*$ are nonzero, where $1 \leq p_0 \leq p$. Let the set $\tilde{\mathcal{A}}_n = \left\{ j : 1 \leq j \leq p,\ \hat{\beta}^*_{j,n} \neq 0 \right\}$ denote the nonzero elements picked by the adaptive lasso. Zou (2006) showed that under mild regularity conditions for constant p and $n \to \infty$, then,

- $\lim_{n\to\infty} P(\tilde{\mathcal{A}}_n = \mathcal{A}) = 1$,
- $\sqrt{n}\left(\hat{\boldsymbol{\beta}}^* - \boldsymbol{\beta}^*\right) \xrightarrow{d} \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{C}_{11}^{-1})$,

where $\mathcal{A}$ denotes the true nonzero elements in $\boldsymbol{\beta}^*$, $\mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{C}_{11}^{-1})$ is a normal distribution with zero mean and $\sigma^2 \mathbf{C}_{11}^{-1}$ variance. $\mathbf{C}_{11}^{-1}$ is the upper left $p_0 \times p_0$ matrix of $\mathbf{C}$, where $\frac{1}{n}\mathbf{X}^\top \mathbf{X} \to \mathbf{C}$ and $\mathbf{C}$ is a positive definite matrix (Chatterjee and Lahiri, 2013)(Zou, 2006).

Therefore, the adaptive lasso enjoys oracle properties (Fan and Li, 2001), meaning that it can correctly identify the set of nonzero components of $\boldsymbol{\beta}^*$ with a probability converging to one. Furthermore, these nonzero components can be estimated correctly, with the same precision as ordinary least squares method (Chatterjee and Lahiri, 2013).

An additional benefit of the adaptive lasso is that it remains a convex optimisation problem. Therefore, its global minimiser can be efficiently solved using a convex optimisation solver (Zou, 2006).

3.7 Cross-validation

The value of λ controls the amount of shrinkage that is applied to the estimates. To determine λ, it is tested across a range of values, for which each fit must be assessed by using k-fold cross-validation. The concept of k-fold cross-validation is to partition the data into k random sets, or folds of approximately equal size. The first fold is withheld and the remaining $k-1$ folds are used for training the algorithm. The mean squares error (MSE) is then computed on the held-out fold. This procedure is repeated k times such that every fold is used once for testing, resulting into k estimates of the error. The k-fold cross-validated error is then computed by averaging these values (James et al., 2013).

An important advantage of k-fold cross-validation is that it gives accurate estimates of the test errors due to its beneficial bias-variance trade-off. When performing k-fold cross validation with $k = n$, or leave-one-out-cross-validation (LOOCV), it leads to an unbiased estimate of the test error as there are $n-1$ observations. Therefore, using 5- or 10-fold cross-validation results in a more biased test error. However, bias is not the only performance indicator of concern. It turns out that LOOCV has higher variance than k-fold cross-validation where $k < n$, due to the high positive correlation between the n fitted models. In contrast, using k-fold cross-validation we average k models which are less correlated with each other, since there is a smaller overlap between test and training data in each model (James et al., 2013). Therefore, 5- or 10-fold cross-validation is typically used as it has a good trade-off between estimation bias and variance (Van Le, 2020).

Chapter 4

Simulation Experiments

In this chapter the performance of the proposed algorithms is assessed for system identification using illustrative numerical examples. Experiments are performed to provide an answer to the following questions:

- What is the effect of varying the sample size on the performance of the system identification?
- Under what conditions is adaptive lasso better for system identification in terms of parameter estimation accuracy than least squares?
- What the effect of link failure during intervention?
- How is a link failure detected without performing system identification for the entire network?

4.1 Model Set-up

The network game used for simulations is based on the numerical example from Shakarami et al. (2021). This is a six player network game, hence $\mathcal{I} \in \{1, 6\}$, which has an interaction network defined by the directed graph visually presented in Figure 4.1. Here, the number in the directed arrows of the graph represent the weight of that link, denoting the influence of player i on player j. The number next to the nodes denotes the standalone marginal return of that player. Furthermore, the pay-off parameter $a = -0.5$ means that the game exhibits strategic substitutes. This choice is without loss of generality, as a positive pay-off parameter changes the type of network game for which it exhibits strategic complements. The initial strategy played by an agent is a random value within the interval $\mathbf{x}_0 \in [0, 1]$. Due to the discretisation, a step size of $h = 0.05$ is introduced to determine the rate at which players update their strategy.

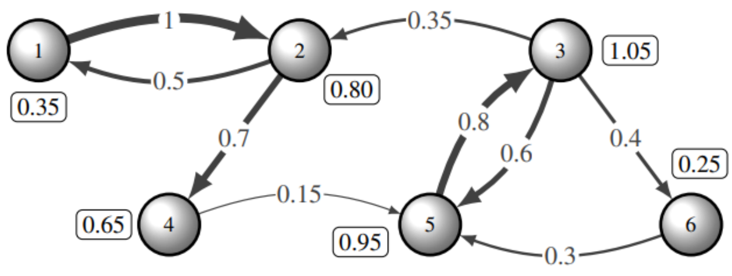


Figure 4.1: The directed graph that defines the interaction network of the network game. Figure retrieved from Shakarami et al. (2021).

4.2 Experimental Set-up

In this sections, experiments are set-up to test the performance of the adaptive lasso for system identification and provide an answer to the research questions at the beginning of this chapter.

4.2.1 Varying sample size

During this experiment, the sample size is varied to determine its effect on the adaptive lasso algorithm. From Equation 3.13 we know that our mathematical model is a linear map that consists of a matrix that contains the network topology, concatenated with a vector containing the marginal return. The network game used for simulation consists of six players. Therefore, Equation 3.13 consists of a 6×6 matrix concatenated with a 6×1 vector, resulting in a 6×7 linear map. As a result, to solve Equation 3.12 at least seven samples are required such that the system is not underdetermined. Therefore, the minimum number of samples for this experiment is seven. As for the varying part of the experiment: the number of samples is doubled five times which means that the algorithm is run for six different sample sizes denoted by $N \in \{7, 14, 28, 56, 112, 244\}$.

For every sample size, the identification algorithm is tested using different noise parameter values. The noise variables are picked from a uniform distribution where the level of noise is determined by the bounds of the distribution. The bounds are set to -1 and 1, meaning that a variable picked from this distribution can take any value between $[-1, 1]$. In the simulation, the level of noise is scaled by a parameter that effectively scales the bounds of the uniform distribution. This noise parameter is denoted by $B_{noise} \in \{0, 0.01, 0.02, 0.05, 0.1\}$.

Multiple parameters of the algorithm are stored during its operation to assess the performance of the algorithm afterwards. These parameters are the maximum payoff obtained from intervention using the identified network parameters, the gamma and lambda value that lead to the best model (highest pay-off) selected by the adaptive lasso algorithm, the l_1 norm of the difference between the equilibrium of the action profile (social optimum) during intervention using identified system parameters and the true social optimum action profile ($\|\mathbf{x}_{opt} - \bar{\mathbf{x}}_{opt})\|_1$). Another parameter is the number of false positive and false negative elements detected by the algorithm. How this parameter is defined will be explained below. Lastly, the computation time of the adaptive lasso is measured and stored for each iteration.

The above named performance indicators are determined for a model given by the least squares solution as well, to compare the performance of the adaptive lasso algorithm to ordinary least squares.

Pay-off parameter

The pay-off or utility obtained by a player is computed using Equation 2.2. Summing this parameter for all players gives the pay-off obtained by the network. This parameter is optimised by applying static intervention to the network using network parameters identified by the adaptive lasso algorithm.

Lambda and gamma

The adaptive lasso algorithm optimises the fit of a model to the data based on the parameters gamma and lambda. Gamma determines the magnitude of the weights used by the adaptive lasso, and lambda determines the amount of regularisation. Therefore, the adaptive lasso finds the best solution on a grid of gamma by lambda. The range and step size of gamma and lambda determine the size and refinement of the grid. Hence, increasing the range of gamma and lambda and decreasing their step size increases the probability of finding the best model that approximates the true model. Gamma is ranged from $\gamma \in \{0, 0.1, 0.2, 0.5, 0.1\}$ and lambda is ranged logarithmically from [1e−6, 1] in 20 steps.

Proximity to social optimum

The pay-off function in Equation 2.2 admits a unique solution which is given by Equation 2.8. For a stable system the players converge to an equilibrium which is denoted by $\bar{x}_{opt}$. The absolute distance of this vector from the true social optimum x_{opt} is defined as the proximity to the social optimum and is used as a performance indicator.

Sparsity index

It is known that the network used for system identification is sparse, therefore, the estimated network parameter $a\mathbf{P}$ must be sparse as well. To assess the sparsity of the identified model, the number of false positive elements are counted. An elements is considered a false positive if its value is greater than a certain threshold value (1e−2) and the element is zero in the true **P** matrix. Furthermore, the number of false negative elements is counted. An element is considered a false negative when it is not within a predefined percentage (30%) of the actual value in the true **P** matrix[1].

Computation time

The computation time is tracked for each iteration of the algorithm to determine the effect of noise and increasing sample size on the computation time of the algorithm.

4.2.2 Adaptive lasso in under- and overdetermined systems

For illustrative purposes the network considered for this thesis is only a six player network game. Therefore, the number of samples quickly exceeds the number of unknown links in the network. However, in practical applications the number of players in a network game may be in the order of magnitude of 10^3. In such settings, the graph that defines the interaction network is 10^3 by 10^3. It is not unthinkable that less than 10^3 number of samples are available, meaning that the regression problem in system identification is underdetermined. If one tries to solve this problem using least squares, a non-sparse solution is found.

[1]As it might occur that the identified element is more than (30%) higher/lower than the actual value of that element it is considered a false negative even though the element is correctly active. Hence, a false negative is not necessarily a false negative by its definition.

Fortunately, there are methods of solving such underdetermined systems of equations using methods that give sparse solutions, such as the adaptive lasso. This experiment is organised to assess the ability of the adaptive lasso algorithm to deliver sparse solutions that approximate the true model in system identification of network games.

As described in the previous experiment, the model that defines the network game consists of six linear equations containing seven unknowns each. Therefore, the exact solution to this regression problem can only be found when there are at least seven samples. For this experiment, the sample size is ranged from two to ten, to see how the adaptive lasso algorithm compares to the least squares solution. The key performance indicators from the previous varying sample size experiment are used. This experiment is repeated using relatively small noise $B_{noise} = 5\text{e}{-5}$, to see its affect on the network parameter estimation accuracy.

4.2.3 Link failure detection

In this section we explore the ability of the system identification algorithm to detect a link failure during intervention. To simulate a link failure, one of elements in the $\mathbf{P}$ matrix is dropped. Consider the graph in Figure 4.1, where we remove the weakest link $P_{54} = 0.15$ during intervention.

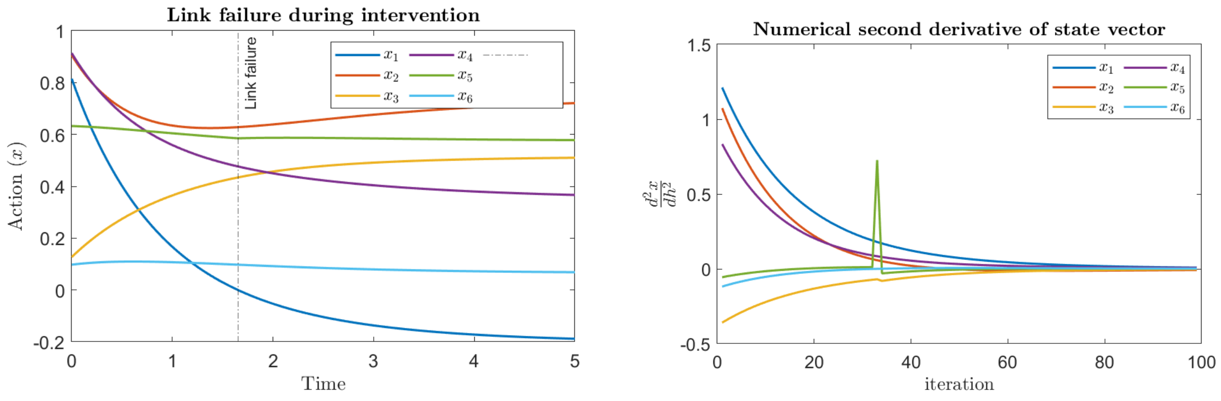
The experiment is performed according to the following steps: first, the ODE that describes the network dynamics receives an input variable $\mathbf{u}(\cdot)$ that contains the exploration signal defined by the central regulator, and a noise variable ϵ. Then, the output data of the ODE (the states of the agents) is stored and the adaptive lasso algorithm is used to identify the underlying interaction network. Using the identified network parameters, a static feedback intervention is applied to the network. At an arbitrary point in time during the applied intervention, a link fails in the network. As a result of this link failure, the action profile that satisfies social welfare will change, and thus, a new suitable intervention must be applied to reach the new social optimum action profile. Therefore, we want the algorithm to detect this link failure, reapply an exploration signal to the changed network, identify the changed network parameters and apply a new suitable intervention that steers the network to the social optimum.

A link failure causes a change in network topology which subsequently affects how the agents interact with each other. This change is noticeable as a discontinuity in the action profile of the agents from the point of the link failure. A discontinuity is a type of irregularity that occurs in an otherwise smooth function, and appears as an abrupt change in direction in the action profile of the agents. Gijbels and Goderniaux (2005) computed the second order derivative of a Nadaraya-Watson kernel estimator to detect a discontinuity in the data. However, as we do not use an estimator, we compute the numerical second order derivative along the player's action profile. This results in a peak at the point where the discontinuity occurs. To understand this effect conceptually, the second order derivative can be thought of as acceleration. When the action profile of an agent changes abruptly, the (negative) acceleration at that point is significant. The numerical second order derivative is defined in Leibniz notation in Equation 4.1.

$$\frac{\mathrm{d}^2 x}{\mathrm{d}h^2} = \begin{bmatrix} \frac{1}{h^2} & \frac{-2}{h^2} & \frac{1}{h^2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}. \tag{4.1}$$

The numerical second order derivative is computed along the action profile of the agents. A discontinuity in the action profile of the agents results in a relative high intensity peak. A peak is detected when the current value of the second order derivative is greater (or smaller) than its subsequent and previous neighbouring data point. A peak is defined as a (local) minimum or maximum. A minimum or maximum occurs when the second order derivative switches from increasing to decreasing (or vice versa) in value, which translates to a switch from an increasing to decreasing value (or vice versa) of the action profile of a player. To correctly identify a peak due to a discontinuity, rather than a switch in action profile of an agent, a small tolerance is used as a threshold value such that relatively small peaks are not falsely assigned to a link failure.

The discontinuity that occurs in the action profile due to the link failure can be observed in figure Figure 4.2a. Even though the discontinuity is difficult to observe visually, it appears as a significant peak in the second order derivative given in Figure 4.2. The figure is used as an illustrative example and is generated for a situation without noise.



(a) Action profile of the agents under static feedback intervention.

(b) Numerical second order derivative computed along the player´s action profile in (a).

Figure 4.2: Subfigure (a) shows the discontinuity in the action profile due to a link failure. Subfigure (b) shows the second order derivative of the action profile in (a). At the point where the link failure occurs, a large peak appears.

The link failure is going to occur at a random iteration. When the algorithm has detected an anomaly in the action profile of any agent(s), system identification must be performed again to accommodate for the change in network topology. However, running the system identification algorithm for the entire network is excessive, thus, a waste of computing power as only a minor proportion of the network structure has changed. Therefore, the identification algorithm should only re-identify the part of the network where a link failure has occurred.

When a link failure has occurred, the agent that is directly affected by that link shows the largest change in action profile. Hence, it is possible to detect which agent is directly affected by the link failure, although, which link has failed cannot be deduced from this information. This means that the system identification algorithm must re-compute the values for each link directed towards that agent. Practically speaking, it means that the elements along the row belonging to that agent in Equation 3.13 must be re-identified and all other elements retain their value from the previous system identification run. The elements of this row describe the influence of all other players on that agent, and his marginal return. After re-identification it is possible to determine which specific link has failed by computing the difference between the elements from before link failure and after link failure.

Chapter 5

Simulation Results

5.1 Results varying sample size

The results of the experiment are presented in Table A.1 in the appendix. The threshold value for when an element is considered a false positive is set at1e−2, meaning that elements in $a\mathbf{P}$ smaller than this threshold are considered zero. An element active in the identified $a\mathbf{P}$ is considered false negative when it is not within 30% of the value in the true $a\mathbf{P}$.

Computation time adaptive lasso

The relation between computation time and sample size for the convergence of the adaptive lasso is important as it may set a limitation on how large the sample size can be. For this experiment the computation time of the adaptive lasso on a lambda by gamma grid was clocked. To obtain more data points, the sample size was doubled twice more to show a better relation between computation time and sample size. The computation time data is plotted in Figure 5.1.

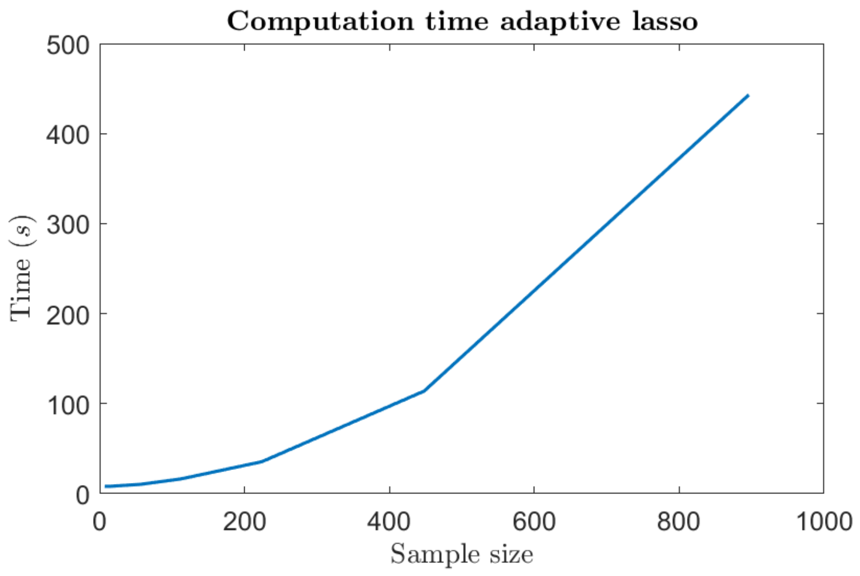


Figure 5.1: This graph shows the relation between computation time and the sample size.

From the graph it is observed that the computation time grows exponentially with respect to the sample size. This imposes a size constraint on the sample size, as adaptive lasso becomes computationally more expensive.

Convergence

The absolute distance of the Nash equilibrium form the social optimum is 1.1053. Any distance closer to the social optimum is considered an improvement. This is due to better estimation of the network parameters such that an adequate intervention can be applied. Using the data from Table A.1 distance from the social optimum is plotted for the adaptive lasso is presented in Figure 5.2.

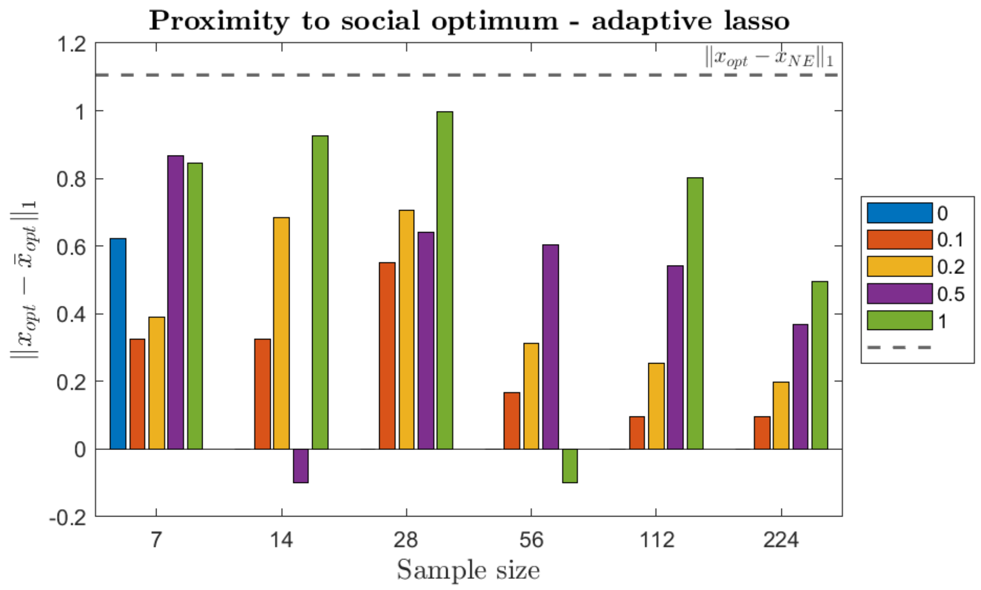


Figure 5.2: A bar plot of the absolute distance from the social optimum for adaptive lasso with respect to the Nash-equilibrium per sample size. The legend shows the noise bounds used in the simulation for system identification. The dashed horizontal line shows the distance of the Nash equilibrium from the social optimum. for unstable interventions, the distance is set to -0.1.

Two main observations are made from Figure 5.2. First, for every noise level, a trend is visible that the distance from the social optimum decreases with increasing sample size, meaning that more samples lead to better estimation of the network parameters and thus, to closer convergence to the social optimum. Second, a more trivial observation is that the distance from the social optimum increases with increasing noise. Thus, for each sample size the five bars that represent the noise levels show an increasing trend.

Another plot presenting the distance from the social optimum for least squares using a bar plot is given in Figure 5.3. The negative values indicate an instability during intervention using the parameters derived from the model fit using least squares.

The trends in Figure 5.3 for least squares are less obvious with respect to adaptive lasso. Nevertheless, it is assumed that the trends described for adaptive lasso hold for least squares. Remarkable is that for some noise and sample size combination, the intervention based on the estimated parameters is unstable. Furthermore, it is observed that for the largest sample size ($N = 224$) the least squares solution converges closer to the social optimum than the adaptive lasso.

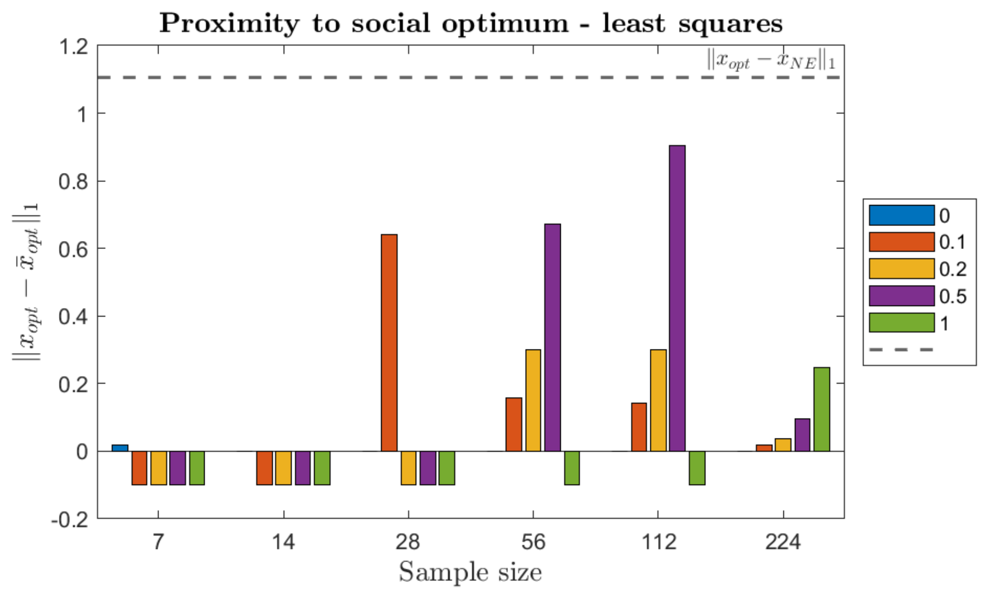


Figure 5.3: A bar plot of the absolute distance from the social optimum for least squares. The legend shows the noise bounds used in the simulation for system identification. The horizontal dashed line shows the distance of the Nash equilibrium from the social optimum. For unstable interventions, the distance is set to -0.1.

Robustness

This large varying sample size experiment showed that the adaptive lasso is more robust than least squares regarding intervention using the identified parameters. The robustness of adaptive lasso is due to the gamma / lambda regularisation, which shrinks elements to zero. When there is much uncertainty due to noise, adaptive lasso selects high gamma/lambda values which heavily regularise the elements in the regression model. Heavy regularisation also affects large elements that are expected to be active in the regression model. Therefore, in a low noise situation, more elements are nonzero than for a high noise situation. Furthermore, in a low noise situation the model fits the data better, even though noise variables are being picked up. In a high noise situation, the heavy regularisation shrinks most elements to zero. Hence, adaptive lasso is more robust than least squares.

Conclusion

The computation time of the adaptive lasso grows exponentially with increasing sample size, imposing a size constraint on the number of samples. Increasing the sample size results in better estimation of the network parameters, thus better convergence to the social optimum for both adaptive lasso and least squares. For lower sample sizes the adaptive lasso gives sparser results and is more robust than least squares, at the cost of reduced prediction accuracy due to the heavy regularisation. When the number of samples is increased further, least squares is outperforming the adaptive lasso in terms of accuracy.

In conclusion; for lower sample sizes, adaptive lasso is preferred over least squares as it is more accurate and robust. When the sample size is large, least squares is a better suited due to its better performance in terms of accuracy and computation time.

5.2 Results adaptive in under- and over-determined systems

The experiment is performed for a sample size N ranging from two to ten, and the noise in the network is zero. The adaptive lasso is computed on a grid of gamma $\gamma \in \{0, 0.1, 0.2, 0.5, 0.1\}$ by lambda , where lambda ranged from 1e$-$5 to 1e0 in 20 steps logarithmically. The performance of the algorithm is compared to least squares based on the maximum pay-off achieved from intervention using the identified network parameters, the l_1 norm of the difference between the social optimum and the true social optimum and the number of false positive and false negative estimated elements. The results are presented in Table 5.1.

	Adaptive Lasso Algorithm						Least Squares			
N	γ	λ	Pay-off	$\lVert x_{opt} - \bar{x}_{opt}\rVert_1$	FP	FN	FP	FN	Pay-off	$\lVert x_{opt} - \bar{x}_{opt}\rVert_1$
2	0.5	4.83e$-$2	0.8736	1.0465	5	8	27	9	0.7250	1.7459
3	0.5	8.86e$-$2	0.9068	0.5907	11	8	27	9	0.7366	1.7116
4	0.5	8.86e$-$2	0.9032	0.7538	11	8	27	9	0.7854	1.6672
5	0.5	3.36e$-$5	0.9169	0.0310	7	3	27	9	0.8486	1.2747
6	0.5	1.00e$-$5	0.9148	0.2364	21	5	27	8	0.9127	0.5792
7	0.5	1.83e$-$5	0.9169	0.0189	0	0	0	0	0.9169	0.0186
8	0.5	1.00e$-$5	0.9169	0.0104	0	0	0	0	0.9169	0.0107
9	0.5	1.00e$-$5	0.9169	0.0060	0	0	0	0	0.9169	0.0062
10	0.05	1.00e$-$5	0.9169	0.0018	0	0	0	0	0.9169	0.0036

Table 5.1: The performance indicators for the adaptive lasso algorithm and least squares solution.

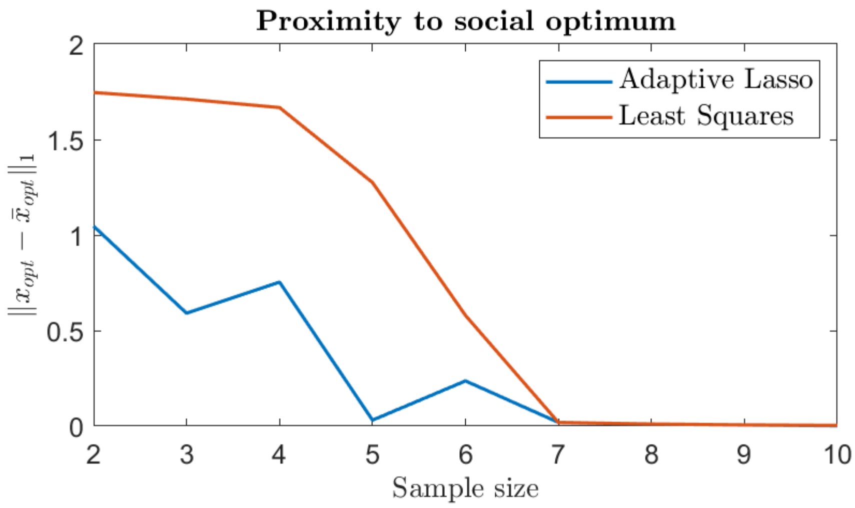


Figure 5.4: This graph shows the convergence to the social optimum for adaptive lasso and least squares for under- and overdetermined systems. There is no noise present in the network.

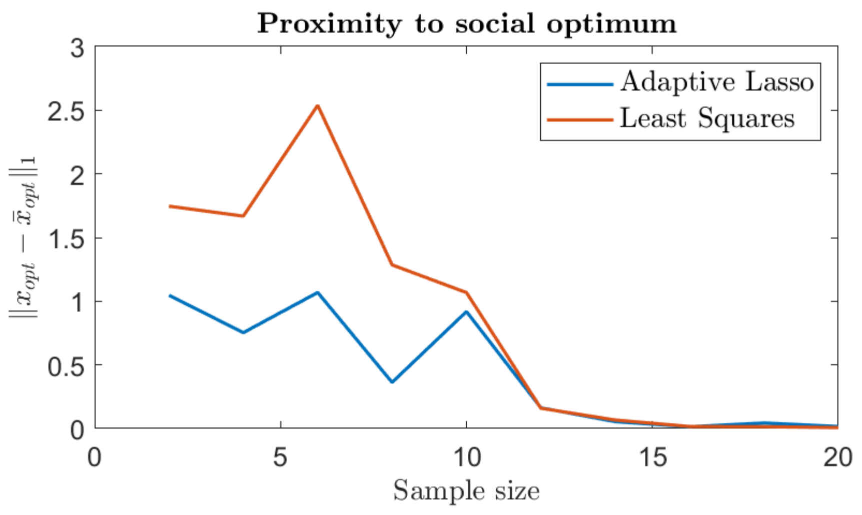


Figure 5.5: This graph shows the convergence to the social optimum for adaptive lasso and least squares for under- and overdetermined systems. Here a small noise of $B_{noise} = 5\text{e}{-5}$ is present in the network.

Interpretation

As the sample size increases, the identified model approximates the true model more accurately which is observed as a decreasing difference between the actual and true social optimum action profile in Figure 5.4. This effect is observed for both the adaptive lasso algorithm and least squares. However, adaptive lasso gives better results when the system is underdetermined which is until the sample size reaches 7. From this point the linear regression is no longer underdetermined and least squares can give the exact solution which is sparse.

However, Figure 5.5 shows that when noise is introduced, the number of samples required to obtain an accurate estimate of the network parameters is increased. From sample size $N = 12$ the distance from the social optimum for adaptive lasso coincides with the distance for the social optimum for least squares, indicating that both estimates of the network parameters are equal.

From the columns presenting the falsely negative predicted elements (7 and 9), it is observed that for the first three iterations, 8 out of 9 elements are falsely identified. When the sample size increases, the adaptive lasso gives less false negatives than least squares. Least squares estimates nearly all elements wrong until $N = 7$. The number of false positives is used to indicate the sparsity of the estimated network parameter. If an element is larger than a threshold value and it is zero in the true model, it is registered as a false positive. Least squares predicts nonzero elements alone until $N = 7$. The number of false positives for the adaptive lasso is growing as the sample size increases. This means that more elements are predicted as nonzero, which seems counter intuitive at first sight. However, as the sample size increases, the regularisation parameter lambda is decreasing, indicating that the amount of regularisation applied by the adaptive lasso is reduced. This explains that it is probable that for a low sample size and high regularisation, the elements of the model are shrunken to zero. As the sample size increases and the amount of regularisation decreases, more elements are picked up to increase the fit of the model.

Conclusion

The adaptive lasso performs better when the system is underdetermined, which is when the number of samples is less than the number of unknowns in linear regression. When the number of samples is growing and the system becomes overdetermined, the least squares estimates coincide with the adaptive lasso. The point where they coincide depends on the level of noise in the network; when the level of noise increases, more samples are required.

5.3 Results link failure

Multiple simulations are performed to assess the performance of the algorithm under noisy conditions. Table 5.2 shows the outcome of the algorithm for multiple simulation runs using different noise parameters $B_{noise} \in \{0, 0.01, 0.02, 0.05, 0.1\}$. The gamma value that determines the weights for the adaptive lasso are ranged from $\gamma \in \{0, 0.1, 0.2, 0.5, 0.1\}$ and the lambda value that determines the amount of regularisation is ranged from 1e−6 to 1e0 in 20 steps logarithmically. The number of samples used is $N = 100$. The link with lowest magnitude $P_{54} = 0.15$ is dropped at iteration 23. The maximum pay-off obtainable before link failure is 0.9169 and after link failure 0.9329. In the table below, B_{noise} denotes the noise parameters used in the simulation, and the mean and variance of all noise variables are given as well. Here, we use the realised pay-off as a performance indicator. The realised pay-off by intervention before link failure (LF) and after link failure, the tuning parameters gamma and lambda and the iteration at which the link failure is detected. The heading "player" denotes which agent is effected the most by the link failure, thus one of the links directed to that agent has failed.

Noise level			Realised pay-off		Tuning parameters				Link Failure (LF)	
B_{noise}	Mean (μ)	Variance (σ^2)	Before LF	After LF	γ_1	γ_2	λ_1	λ_2	Iteration	Player
0	0	0	0.9169	0.9329	0.05	0.05	1e−6	1e−6	23	5
0.01	−4.4631e−5	3.1200e−5	0.9167	0.9327	0.1	0.1	1e−6	1e−6	23	5
0.02	−8.9262e−5	1.2480e−4	0.9161	0.9327	0.1	0.5	6.9519e−2	1e−6	23	5
0.05	−2.2315e−4	7.8001e−4	0.9136	0.9298	0.1	0.5	2.9764e−3	1e−6	23	5
0.1	−4.4631e−4	3.1200e−3	0.9079	0.9203	0.2	0.05	1e−6	2.0691e−6	23	5

Table 5.2: Simulation results of the link failure identification algorithm.

Interpretation

From Table 5.2 it is observed that for the no noise situation, the maximum pay-off is achieved before and after link failure, and that the correct iteration of link failure and agent are identified. This indicates that the algorithm behaves as expected. With increasing noise, the realised pay-off is decreasing. This indicates that the estimation of the parameters used for intervention have picked up noise variables. Nevertheless, the link failure is detected at the correct iteration and for the player that is directly affected.

Table 5.3 shows the values assigned to each element that denotes a link directed towards player five, before and after link failure, for different noise parameters. In the zero noise situation,

$B_{noise} \downarrow$		P_{51}	P_{52}	P_{53}	P_{54}	P_{55}	P_{56}
0	Before LF	0	0	-0.3000	-0.0750	0	-0.1500
	After LF	0	0	-0.3000	0	0	-0.1500
	Difference	0	0	0	0.0750	0	0
0.01	Before LF	0.0017	0.0045	-0.2878	-0.0701	-0.0104	-0.1506
	After LF	-0.0008	0.0009	-0.2840	0.0028	0.0091	-0.1506
	Difference	0.0025	0.0034	0.0038	0.0729	0.0195	0.0000
0.02	Before LF	0.0017	0.0051	-0.2756	-0.0662	-0.0185	-0.1496
	After LF	-0.0006	0.0133	-0.3253	0.0022	0.0107	-0.1471
	Difference	0.0023	0.0082	0.0497	0.0684	0.0292	0.0025
0.05	Before LF	0.0029	0.0075	-0.2394	-0.0519	-0.0411	-0.1442
	After LF	-0.0035	0.0006	-0.2118	0.0196	0.0430	-0.1515
	Difference	0.0064	0.0069	0.0276	0.0715	0.0841	0.0073
0.1	Before LF	0.0040	0.0105	-0.1800	-0.0232	-0.0720	-0.1279
	After LF	-0.0005	0.0028	-0.1155	0.0537	0.0793	-0.1461
	Difference	0.0045	0.0077	0.0645	0.0769	0.1513	0.0182

Table 5.3: The link strengths before and after link failure identified by the link failure detection algorithm. The exact link strengths are found in the no noise situation ($B_{noise} = 0$).

all links are correctly identified before and after link failure. As the noise increases, some noise variables are picked up, resulting in more nonzero elements. However, considering that the picked up noise variables are small in magnitude, it is observed that the algorithm can correctly identify the link failure. The failing link is assigned to the element with the absolute largest difference before and after link failure. The first three runs correctly identify P_{54} as the failing link. In the last two runs, P_{55} has the largest change in magnitude and is falsely detected as the failing link.

Conclusion

As long as the noise level in the network remains below a certain value (in this case 0.05), noise variables that have been picked up by the algorithm are not significant enough to obstruct the identification of a link failure. For this experiment, the weakest link in magnitude was dropped, which was identified in three out of five simulation runs. The algorithm misidentified the link failure for the last two iterations.

Chapter 6

Conclusion

In this thesis, we considered system identification of quadratic network games based on data. The players in the network game choose their actions continuously using pseudo-gradient dynamics, maximising their individual pay-off. By observing these actions over time, large data matrices can be composed containing the dynamics of the agents. Multiple algorithms were proposed to analyse this data and extract the network parameters that define the network game. The network parameters are used by a central regulator to apply interventions to the network and steer the players to a social optimum. The performance of the network parameter estimation by the proposed algorithm was assessed by how close the network is steered towards the social optimum.

The algorithm used for system identification is based on the adaptive lasso. This is a linear regression technique that applies regularisation to promote sparsity in the solution. Its performance was tested using two experiments where the number of samples and level of noise in the network was varied. The experiments showed that adaptive lasso is robust in situations where the number of samples is relatively low, or the noise level is relatively high. The second algorithm used for system identification is the least squares solution computed using a singular value decomposition. This is a fast method that does not give sparse solutions to underdetermined systems. However, when the number of samples is increased, it can identify the true active elements correctly.

A drawback of the adaptive lasso is that it requires optimisation of two parameters, increasing the computational cost. Furthermore, the computation time scales exponentially with increasing sample size limiting the practical application of the lasso. However, when the sample size is large and the system is overdetermined, the least squares solution outperforms the adaptive lasso, and is significantly faster to compute. Therefore, alleviating the scaling problem of the adaptive lasso. When the system is underdetermined, the adaptive lasso is robust and estimates the network parameters surprisingly good, in contrast to the least squares solution. Therefore, adaptive lasso is preferred in situations where the number of samples is less than the number of players in the network.

The third experiment showed that the adaptive lasso is capable of identifying the player that is affected the most by a link failure under noisy conditions. The re-identification is only performed for the links directed towards the agent that is effected the most. By comparing the newly identified links to the old links, it is determined which exact link had failed. When the noise level becomes significant, elements are picked up due to noise which obstructs the correct identification of the link failure.

Chapter 7

Discussion

The (pseudo) random numbers generated for the experiments are drawn from a distribution using one seed value, which restricts the statistical significance of the results from the experiments. Increasing the number of replications using different seed values increases the statistical significance and would show the trends in the data more clearly.

The adaptive lasso is superior to least squares when the regression problem performed is underdetermined. When there is noise present in the system, the adaptive lasso outperforms least squares in terms of accuracy, although, this difference in performance decreases when the number of samples is increased. As least squares is significantly less computationally expensive, it is recommended to collect more samples and compute the least squares solution, than relying on the sparsity promoting characteristic of the adaptive lasso in underdetermined systems. Naturally, it depends on the practical application whether this choice is available or not.

Furthermore, least squares does not give sparse solution when the sample size is increased, however, the estimate of the true active elements converges to their true value. The zero elements have picked up minor noise variables. The weights computed for the adaptive lasso depend on an initial estimate of the model. During the research the weights were computed using a lasso estimate with regularisation parameter $\lambda = 1\text{e}{-3}$. When the least squares estimate is close to the true solution, it makes a better initial estimate to base the adaptive weights on, such that the true active elements remain active in the solution of the adaptive lasso, and the zero elements that have picked up noise, are shrunken to zero.

In this research it was observed that when the number of samples increases, the estimation of the network parameters converges to the true value of the network parameters. However, no conditions are given on the number of samples that are minimally required to estimate the true model. Another limitation of the research is that the performance of the algorithm is assessed based on convergence of the network to the social optimum. This is done by applying interventions based on the estimated network parameters. However, this type of assessment requires knowledge of the true network parameters, which might not be accessible in real applications.

Finally, it was observed that when a significant level of is noise present in the network with respect to the number of samples, the adaptive lasso shrunk predictor elements to zero to minimise the objective function. This means that such heavy regularisation increased the weight of the l_1 norm significantly with respect to the weight of the l_2 norm of the minimisation algorithm.

7.1 Future Work

Future research is required to solve the limitations pointed out in the discussion. For example, future research can explore the relationship between prediction accuracy and sample size, providing stricter criteria on how many samples are minimally required to obtain an accurate estimate of the network parameters. Furthermore, it could explore the possibility to assess the performance of the system identification algorithm based on cross validation as explained in section 3.7.

To increase the weight of the l_2 norm, a regularisation parameter can be added to this norm to increase its importance with respect to the l_1 norm. This can have the effect that the error between the model and data is more balanced with respect to the size constraints on the predictor variables in the l_1 norm Future research is required to investigate if such a regularisation parameter on the l_2 norm leads to a better estimation of the network parameters in noisy situations.

Finally, the research can be extended by exploring the performance of the adaptive lasso in larger network games to see how it compares to least squares. Furthermore, the system identification algorithm can be extended to identify the network parameters for network games that have constraints imposed on a subset of players. As such constraints lead to abnormal behaviour in the action profile of those players with respect to unconstrained players.

Bibliography

Alpcan, T., Pavel, L., and Stefanovic, N. (2009). A control theoretic approach to noncooperative game design. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 8575–8580. IEEE.

Bramoullé, Y. and Kranton, R. (2015). Games played on networks. *The Oxford handbook of the economics of networks.*

Brunton, S. L. and Kutz, J. N. (2019). *Data-driven science and engineering: Machine learning, dynamical systems, and control.* Cambridge University Press.

Chatterjee, A. and Lahiri, S. N. (2013). Rates of convergence of the adaptive lasso estimators to the oracle distribution and higher order refinements by the bootstrap. *The Annals of Statistics*, 41(3):1232–1259.

Colin, K., Bombois, X., Bako, L., and Morelli, F. (2019). Informativity: how to get just sufficiently rich for the identification of miso fir systems with multisine excitation? In *2019 18th European Control Conference (ECC)*, pages 351–356. IEEE.

De Persis, C. and Grammatico, S. (2019). Distributed averaging integral nash equilibrium seeking on networks. *Automatica*, 110:108548.

Donoho, D. L. (2006). For most large underdetermined systems of linear equations the minimal l-norm solution is also the sparsest solution. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 59(6):797–829.

Engwerda, J. (2005). *LQ dynamic optimization and differential games.* John Wiley & Sons.

Engwerda, J. C. (2006). Linear quadratic games: An overview. *CentER Discussion Paper Series.*

Fan, J. and Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American statistical Association*, 96(456):1348–1360.

Galeotti, A., Goyal, S., Jackson, M. O., Vega-Redondo, F., and Yariv, L. (2010). Network games. *The review of economic studies*, 77(1):218–244.

Gevers, M., Bazanella, A., and Miskovic, L. (2008). Informative data: how to get just sufficiently rich? In *2008 47th IEEE Conference on Decision and Control*, pages 1962–1967. IEEE.

Gijbels, I. and Goderniaux, A.-C. (2005). Data-driven discontinuity detection in derivatives of a regression function. *Communications in Statistics-Theory and Methods*, 33(4):851–871.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An introduction to statistical learning*, volume 112. Springer.

Koutsoupias, E. and Papadimitriou, C. (1999). Worst-case equilibria. In *Annual symposium on theoretical aspects of computer science*, pages 404–413. Springer.

Ljung, L. (1998). System identification. In *Signal analysis and prediction*, pages 163–173. Springer.

Nagurney, A. (2021). Supply chain game theory network modeling under labor constraints: Applications to the covid-19 pandemic. *European Journal of Operational Research*, 293(3):880–891.

Nagurney, A. and Li, D. (2015). A supply chain network game theory model with product differentiation, outsourcing of production and distribution, and quality and price competition. *Annals of Operations Research*, 226(1):479–503.

Parise, F. and Ozdaglar, A. (2021). Analysis and interventions in large network games. *Annual Review of Control, Robotics, and Autonomous Systems*, 4:455–486.

Shakarami, M., Cherukuri, A., and Monshizadeh, N. (2021). Adaptive interventions for social welfare maximization in network games. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 942–947. IEEE.

Su, W., Bogdan, M., and Candes, E. (2017). False discoveries occur early on the lasso path. *The Annals of statistics*, pages 2133–2150.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288.

Tropp, J. A. (2006). Just relax: Convex programming methods for identifying sparse signals in noise. *IEEE transactions on information theory*, 52(3):1030–1051.

Van Le, C. (2020). How to choose tuning parameters in lasso and ridge regression? *Asian Journal of Economics and Banking*.

Zou, H. (2006). The adaptive lasso and its oracle properties. *Journal of the American statistical association*, 101(476):1418–1429.

Appendix A

Data Matrix Sample Size Variation

		Adaptive Lasso Algorithm						Least Squares				
N	B_{noise}	Time	γ	λ	Pay-off	$\Vert x_{opt} - \bar{x}_{opt}\Vert_1$	FP	FN	FP	FN	Pay-off	$\Vert x_{opt} - \bar{x}_{opt}\Vert_1$
7	0	8.07	0.05	1.00e−5	0.9122	0.6220	22	2	0	0	0.9169	0.0186
	0.01	7.55	0.05	1.44e−2	0.9122	0.3264	7	4	27	9	-1	-1
	0.02	8.49	0.1	8.86e−2	0.9031	0.3885	9	6	27	9	-1	-1
	0.05	8.27	0.05	2.98e−1	0.8971	0.8655	8	8	27	9	-1	-1
	0.1	8.76	0.1	8.86e−2	0.8991	0.8439	5	8	27	9	-1	-1
14	0	8.08	0.2	1.00e−5	0.9169	0.0004	0	0	0	0	0.9169	0.0004
	0.01	8.05	0.05	1.27e−3	0.9024	0.3239	19	4	27	8	-1	-1
	0.02	8.13	0.2	6.95e−3	0.9098	0.6855	24	5	27	8	-1	-1
	0.05	8.32	0.2	6.95e−3	0.8966	1.1658	20	6	27	8	-1	-1
	0.1	8.37	0.05	5.56e−1	0.8682	0.9259	6	8	27	8	-1	-1
28	0	8.8	0.05	3.36e−5	0.9169	0.0000	0	0	0	0	0.9169	0.0000
	0.01	8.93	1	2.34e−3	0.9012	0.5496	12	3	24	2	0.8994	0.6415
	0.02	8.98	1	4.28e−3	0.9018	0.7072	10	7	26	3	0.8166	1.5383
	0.05	9.04	0.5	7.85e−3	0.9104	0.6420	12	5	27	5	-1	-1
	0.1	8.91	0.05	2.98e−1	0.8973	0.9973	8	8	27	7	-1	-1
56	0	10.11	0.05	1.13e−4	0.9169	0.0000	0	0	0	0	0.9169	0.0000
	0.01	10.21	0.2	3.79e−4	0.9166	0.1678	9	0	13	0	0.9166	0.1566
	0.02	10.31	0.2	2.34e−3	0.9157	0.3137	16	0	23	0	0.9156	0.3005
	0.05	10.53	1	6.16e−5	0.9115	0.6023	10	3	24	1	0.9095	0.6710
	0.1	10.58	1	2.07e−4	0.8981	1.1031	12	7	26	3	0.8899	1.1420
112	0	15.01	0.05	1.00e−5	0.9169	0.0000	0	0	0	0	0.9169	0.0000
	0.01	16.44	0.05	4.83e−2	0.9168	0.096	6	0	7	0	0.9167	0.1417
	0.02	16.79	0.05	1.62e−1	0.9161	0.2544	8	0	12	0	0.9158	0.3004
	0.05	16.38	0.05	1.62e−1	0.9131	0.5406	14	0	23	0	0.9064	0.9046
	0.1	17.2	0.05	1.62e−1	0.9078	0.8017	13	4	26	1	0.8269	2.5800
224	0	34.18	0.1	1.00e−5	0.9169	0.0000	0	0	0	0	0.9169	0.0000
	0.01	36.15	0.05	2.64e−2	0.9168	0.0968	4	0	1	0	0.9169	0.0175
	0.02	35.65	0.05	4.83e−2	0.9163	0.1984	9	0	9	0	0.9168	0.0354
	0.05	35.61	0.05	1.44e−2	0.9143	0.3698	15	0	17	0	0.9162	0.0949
	0.1	36.06	0.05	6.95e−4	0.9093	0.4959	19	5	21	0	0.9139	0.2487

Table A.1: The performance indicators for the adaptive lasso algorithm and least square solution.

Table A.1 contains data of the performance indicators used to assess the performance of adaptive lasso and least squares. The first column contains the sample sizes, the second column the time to compute the adaptive lasso solution on a gamma by lambda grid. The total pay-off and l_1 norm deviation from the social optimum are given in column 5,6 and 11,12 respectively. FP and FN denote the number of false positive and false negative elements detected by adaptive lasso or least squares.

For some models identified using least squares, the network diverged during intervention, or converged to equilibria unattainable due to a violation of Equation 2.7. For those situations, the value is set to negative one.