Imperial College London
Department of Mathematics

Smarter Networks, Sharper Forecasts; Improving GNAR Prediction Accuracy by Network Search

Harry McCarthy

CID: 02315543

Supervised by Prof Guy Nason

August 29, 2024

Submitted in partial fulfilment of the requirements for the MSc in Statistics at Imperial College London

The work contained in this thesis is my own work unless otherwise stated.

Signed: Harry McCarthy Date: 29/08/2024

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Prof. Guy Nason, for his invaluable expertise and guidance throughout this project. I also extend my thanks to Dr. Pike-Burke for her assistance with Thompson Sampling and to Henry Antonio Palasciano for his insightful advice on forecasting network time series.

Abstract

This thesis is centred around improving predictive performance for Generalized Network Autoregressive (GNAR) models through advanced network search techniques. GNAR models are designed to integrate multivariate time series with an underlying network structure, which captures the critical relationships between time series that are essential for accurate predictions. However, identifying these network structures poses a significant challenge. To address this, we propose novel adaptations of Thompson Sampling and Genetic Algorithms to search for networks which optimise GNAR predictive accuracy. The performance of these methods is evaluated using within-sample predictive accuracy as the objective function. Our results demonstrate that both Thompson Sampling and Genetic Algorithms surpass a simple heuristic-based approach across all model configurations. Further, the Genetic Algorithm achieved exceptional performance, yielding near-zero prediction error and significantly outperforming comparable approaches. Conversely, Thompson Sampling frequently fails to outperform brute force methods, suggesting potential serial dependencies among network edges. These findings highlight the effectiveness of Genetic Algorithms in optimizing network structures for GNAR models and provide insights into the limitations of Thompson Sampling in this context. We apply these methodology to an OECD member state monthly unemployment rate dataset, from the period 1 January 2000 – 1 December 2020.

1. Introduction

In today's rapidly evolving world, where advancements in artificial intelligence run concurrently with elevated inflation rates, ongoing global conflicts and mass migration, accurately forecasting international unemployment rates remains a key element effective economic policies.

With this in mind, Generalized Network Autoregressive (GNAR) models (Knight et al., 2019, 2016; Zhu et al., 2019), have proven to be invaluable tools for forecasting nodal time series data. These models offer a parsimonious and interpretable framework, often outperforming alternative approaches, even in high dimensional settings.

GNAR models achieve this superior performance by coupling a multivariate time series with a network structure that identifies which relationships between the time series are relevant for prediction. Popular alternative models, such as Vector Autoregressive (VAR) models, treat all variables equally, not accounting for any underlying network structure. Consequently, these models are vulnerable to the well-known *curse of dimensionality*. GNAR models circumvent this issue by leveraging the underlying network structure between the time series to estimate only the relevant parameters.

Modifications to the vanilla GNAR model have been made in the past have yielded some success. Mantziou et al. (2023) introduced the **GNAR-edge** model, which allows for time varying edge weights and found improved predictive performance on on a UK business payments dataset. Further, Nason and Wei (2022) proposed the **GNARX** which extends the GNAR model to include node-specific exogenous variables, they found this extension improved upon the original models ability to quantify the economic response to COVID-19 restrictions and death rates.

Returning to the crux of this thesis, GNAR models crucially rely on an input network structure which describes the connectivity between variables of the time series. This network structure can be represented mathematically as a graph consisting of nodes and edges. Here, nodes correspond to the time series values of different variables, and edges indicate the connections between these nodes. However, determining meaningful network structures is a non-trivial exercise. Some researchers have leaned on domain knowledge to construct these networks, Nason and Wei (2022). Others have adopted a brute force approach, producing many networks randomly and opting for the best performing network amongst that cohort, Knight et al. (2019). Performance, in this context, is measured by the network's ability to maximize within-sample prediction accuracy, with the chosen network subsequently used for out-of-sample predictions. In other words, the network which most accurately predicts the most recent of the *observed* data points is the one which is chosen to forecast future, *unobserved* data. This performance evaluation method is straightforward and unbiased, as it simply relies on the parameterisation that performs best up to the current point. Making it our *best guess* at the true underlying network structure.

In contrast to the previous improvements to the standard GNAR model mentioned above, in this thesis our objective is to provide a more advanced and objective methodology for the network discovery process, rather than make changes to the model itself. With careful adaptation, we apply a Bayesian approach through Thompson Sampling along with a metaheuristic Genetic Algorithm to address this problem. The performance of the solutions generated by these two distinct algorithms is compared using the GNAR model's within-sample next-step-ahead prediction error.

In the absence of a known or inferred input network, methods have been proposed to learn the underlying network structure. One approach researchers have taken in the past is to form networks based off correlations between the time series included. Friedman et al. (2007) proposed the *graphical lasso* algorithm that introduces a $L1$-penalty term to the estimation of the inverse covariance matrix, Σ^{-1}. They highlight that if the ijth entry of Σ^{-1} is zero, then variables i and j are conditionally independent of each other, implying no connection or edge exists between them. Hence, by penalising non-zero entries of Σ^{-1}, they could produce a sparse, interpretable network - capturing only the most significant relationships between variables networks.

Alternative approaches have focused on *Granger Causal* (Granger, 1969) relationships to learn graphical structures. Arnold et al. (2007) introduce an algorithm to construct directed network based on whether one time series *Granger-causes* another. The key idea underpinning Granger causality is that if we have time series variables X and Y, where the lags both Y and X are better at predicting Y than the lags of Y alone, then X is said to *granger cause* Y.

Other researchers (Chan-Lau, 2017; Diebold and Yılmaz, 2014) have adopted a variance decomposition angle, where they attempt to quantify how much of the variance in the forecast error of a variable can be attributed to other variables in the system. The premise here is that this decomposition metric measures the influence of one variable on another and thus indicates connectedness.

Our methods deviate from past research in that, rather than approximating a network structure based on properties of the underlying time series. We use within-sample prediction accuracy as an objective function and implement algorithmic approaches to design a network that optimises this objective function. We apply our approaches to an OECD member state unemployment dataset spanning the range 1 January 2000 - 1 December 2020.

2. Preprocessing

2.0.1. Software

The entirety of the programming used in this thesis was performed in R. The **GNAR** package was used fit Generalised Network Autoregressive models throughout. This package was introduced in Knight et al. (2019), where its functionality is described in great detail. Another package used extensively in this thesis is the **GA** package, used for implementing Genetic Algorithms, Scrucca (2013). All instances of Thompson's Sampling were coded directly in R without the use of external packages. The code and data used to generate the results in this thesis are available at the following repositories `https://github.com/harrymccarthy99/Thomspon_Sampling_for_GNAR.git` and `https://github.com/harrymccarthy99/Genetic_Algo_for_GNAR`.

2.0.2. Data

The data used in this thesis comprises monthly unemployment rates for 37 OECD member states spanning the period 1 January 2000 to 1 December 2022, obtained from OECD (2024). The majority of the countries in the dataset are European (26), the rest are distributed across Asia (4), North America (4), South America (2) and Oceania (1).

In this thesis, these countries are often referred to using their alpha-2 ISO codes, which are available in a lookup table provided in the Appendix (Section A).

A notable point here is that while the dataset is mostly complete, there are some instances where data are censored. Staring from 1 January 2000, there are five countries that are missing data as illustrated by Figure 1. Additionally, it's worth mentioning that the **GNAR** model conveniently manages missing data by excluding any node with missing data points from the model fit, reducing the need for imputation.

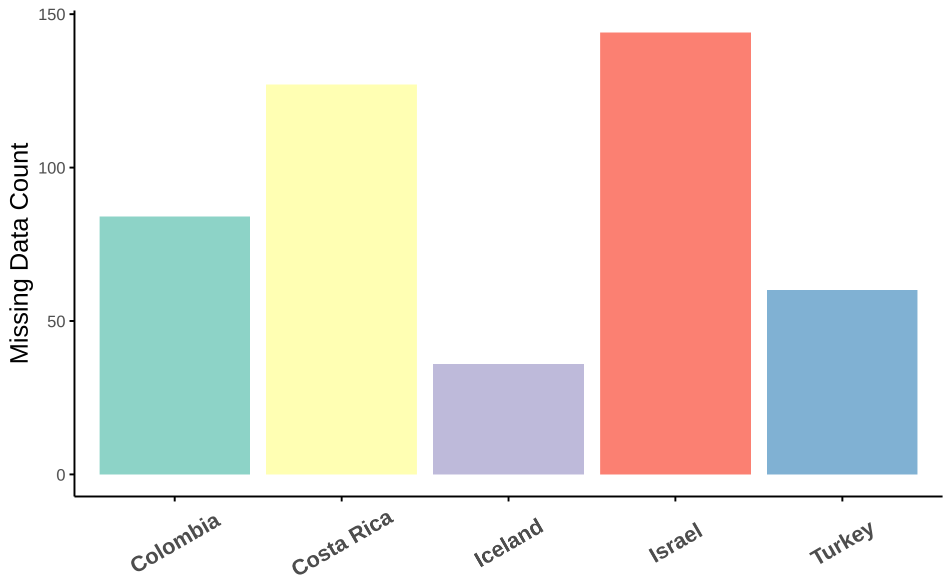

Figure 1.: OECD Unemployment Rate Missing Data Count

2.0.3. Analysis

Figure 2 shows how OECD member state unemployment rates have evolved since the turn of the century. Two notable incidents are highlighted, the first of which is the collapse of former US investment-bank *Lehman Brothers* in Sep 2008, widely regarded as a key accelerant of the Global Financial Crisis (GFC). The second event is the World Health Organization's declaration of COVID-19 as a Public Health Emergency in Jan 2020, marking the onset of the pandemic's severe economic impacts. This plot indicates a great deal of comovement amongst the countries, yet clearly the impact of these economic events varied across nations.

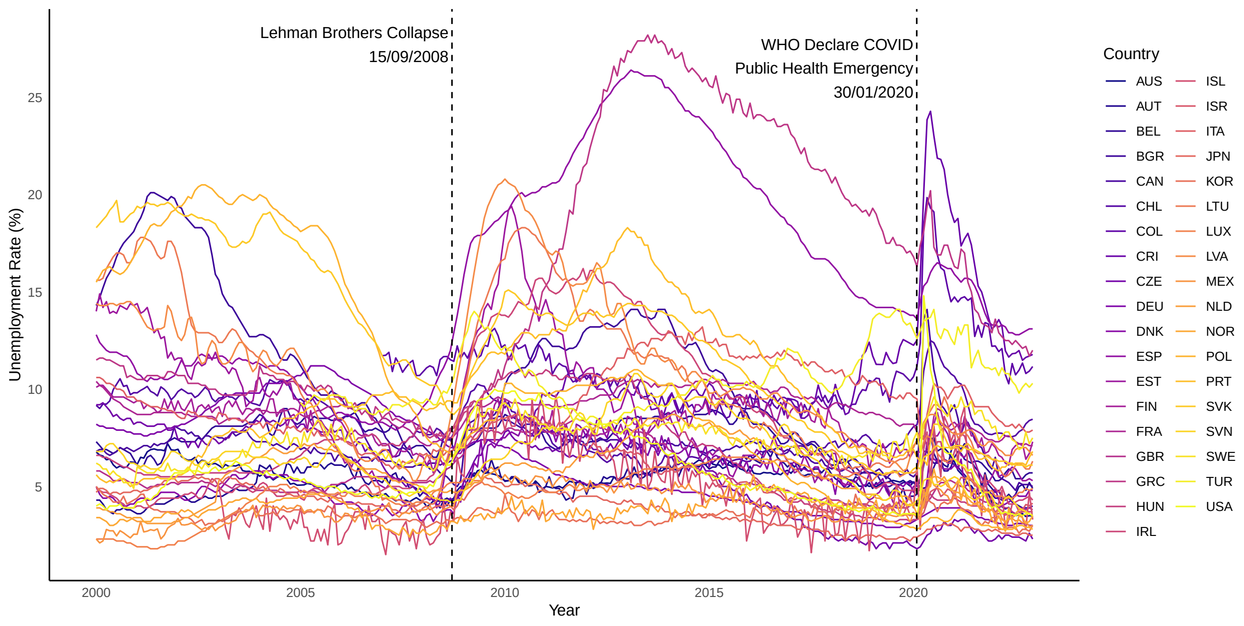

Figure 2.: OECD Member State Unemployment Rates 1 Jan 2000 - 1 Dec 2022

An important note to the reader is that the average monthly difference in unemployment rates is approximately -0.01 and the median monthly difference is approximately -0.02. Hence, given the autoregressive nature inherent in GNAR models, we would anticipate that a well-performing model should yield a low prediction error in this context.

3. Generalized Network Autoregressive Model (GNAR)

3.1. Network Notation

In the following, we follow closely the notation used in Knight et al. (2019) unless otherwise stated.

Network time series are stochastic processes describing the dynamic dependencies between nodes of a multivariate time series. Mathematically they are represented as a pair, $\mathcal{X} = (\boldsymbol{X}_t, \mathcal{G})$, wherein $\boldsymbol{X}_t \in \mathbb{R}^N$ denotes the time series data for the nodes and

$\mathcal{G}$ represents the connections these nodes. The network $\mathcal{G} = (\mathcal{K}, \mathcal{E})$, is composed of the node set $\mathcal{K} = \{1, ..., N\}$ and edge set $\mathcal{E} \subseteq \mathcal{K} \times \mathcal{K}$ which describes the connections between these nodes.

Throughout this thesis, we consider only undirected graphs $\mathcal{G}$, where each edge is between nodes $i, j \in \mathcal{K}$ is bidirectional, denoted as $i \leftrightsquigarrow j$. Consequently, the edge set of a graph with undirected edges is $\mathcal{E} = \{(i, j) : i \leftrightsquigarrow j\}$

The concepts of neighbours and stage-neighbours which are central to this thesis can be described as follows; Letting $A \subset \mathcal{K}$ be a subset of nodes. The stage-1 neighbourhood set of A is denoted by $\mathcal{N}(A)$ and is given by $\mathcal{N}(A) = \{j \in \mathcal{K}/A : i \leftrightsquigarrow j; i \in A\}$. For node $i \in \mathcal{K}$, the rth stage neighbourhood $\mathcal{N}^{(r)}(i)$ is defined as $\mathcal{N}^{(r)}(i) = \mathcal{N}\{\mathcal{N}^{(r-1)}(i)\}/[\{\cup_{q=1}^{(r-1)} \mathcal{N}^{(q)}(i)\} \cup \{i\}]$ for $r = 2, 3, ...$ and $\mathcal{N}^{(1)}(i) = \mathcal{N}(\{i\})$. An alternative interpretation is to say that nodes i and j are r-stage neighbours if the shortest path between them is of length r, in other words, $d(i, j) = r$, Nason et al. (2024).

Each neighbour in a network can have an associated connection weight, $\omega \in [0, 1]$. In our thesis, the edge weight is determined by the size of the r-stage neighbourhood of a node. For example, if node $i \in \mathcal{K}$ has three stage one neighbours i.e. $|N^{(1)}(i)| = 3$, then $\omega_{i,j} = 1/3$ for all $j \in N^{(1)}(i)$.

Another important piece network notation relevant to the GNAR model is that of edge and node covariates. Here, we allow a discrete covariate $c \in C$ to encode a type to an edge or node. For example, Nason et al. (2024) use the these in modelling US presidential elections to encode whether a US state is classed as a *Red*, *Blue* or *Swing* state.

3.2. GNAR Model Overview

The GNAR model is specifically designed to capture the time-varying dependence structures in network time series data. It achieves this by incorporating edge weights determined through both a regular autoregressive component and a neighborhood regression component.

Given a vector of nodal time series data $\mathbf{X}_t = (X_{1,t}, ..., X_{N,t})^T$, the GNAR$(p, [\mathbf{s}])$ model for $\mathbf{X}_t$ for each node $i \in 1, ..., N$ and each time $t \in 1, ..., T$ is given by Equation (1),

$$X_{i,t} = \sum_{j=1}^{p} \left(\alpha_{i,j} X_{i,t-j} + \sum_{c=1}^{C} \sum_{r=1}^{s_j} \beta_{j,r,c} \sum_{q \in \mathcal{N}_t^{(r)}(i)} \right) \tag{1}$$

$$= \underbrace{\sum_{j=1}^{p} (\alpha_{i,j} X_{i,t-j})}_{\text{Autoregression}} + \underbrace{\sum_{j=1}^{p} \left(\sum_{c=1}^{C} \sum_{r=1}^{s_j} \beta_{j,r,c} \sum_{q \in \mathcal{N}_t^{(r)}(i)} \omega_{i,q,c}^{(t)} X_{q,t-j} \right)}_{\text{Neighborhood Regression}} + u_{i,t} \tag{2}$$

Here, p denotes the maximum lag considered and $[\mathbf{s}] = (s_1, ..., s_p)$, where $s_j \in \mathbb{N}$ indicates the extent of the neighbourhood for lag j. The term $\mathcal{N}_t^{(r)}(i)$ denotes the r-stage neighbourhood of node i at time t. Finally, $\omega_{i,q,c}^{(t)}$ represents the connection weight of node i to node q for covariate $c \in C$ where C is the set of covariates under consideration. It is assumed that the noise term for each node $u_{i,t}$ is independently and identically normally distributed for each node i with mean zero and variance σ_i^2

We note that the first expression in Equation (2) can be seen as a standard autoregressive component, where the value of node i is influenced by p lags of itself. The latter term is known as the neighbourhood regression component and is clearly sensitive to the input network structure since we sum over all r-stage neighbours of node i.

A final detail we note is that within the **GNAR** package, there exists functionality to fit either a global-α or local-*alpha* GNAR model. In the globla-α model the same autoregressive coefficient is assumed for each node. In contrast, local-α configurations allow for a unique coefficient for all nodes.

3.3. Connectivity Restrictions

When fitting a GNAR$(p, [\mathbf{s}])$ model, the input network must include at least one s_i-stage neighbor for each $i = 1, \ldots, p$. Since this thesis aims to identify networks that enhance the GNAR predictive accuracy for the OECD unemployment rate dataset, it was necessary to limit our network search to those meeting this s_i-stage neighbor criterion.

To navigate this issue, we propose a base fully-connected network, which we then add edges to in order to improve predictive accuracy. Specifically, we use the minimum spanning tree (MST), where edges are selected to minimise the cumulative distance covered by the network whilst maintaining full connectivity.

Note, that although the MST is designed to minimise distance in the network, the edges we use remain equally distanced. We simply use the MST to decide which edges to include in our network and not to decide how to distance nodes from one another. We base this MST on the distances between the capital cities of the OECD member states included in the study. With this in mind, Figure 3 displays the fully connected network that forms the base of our network exploration.

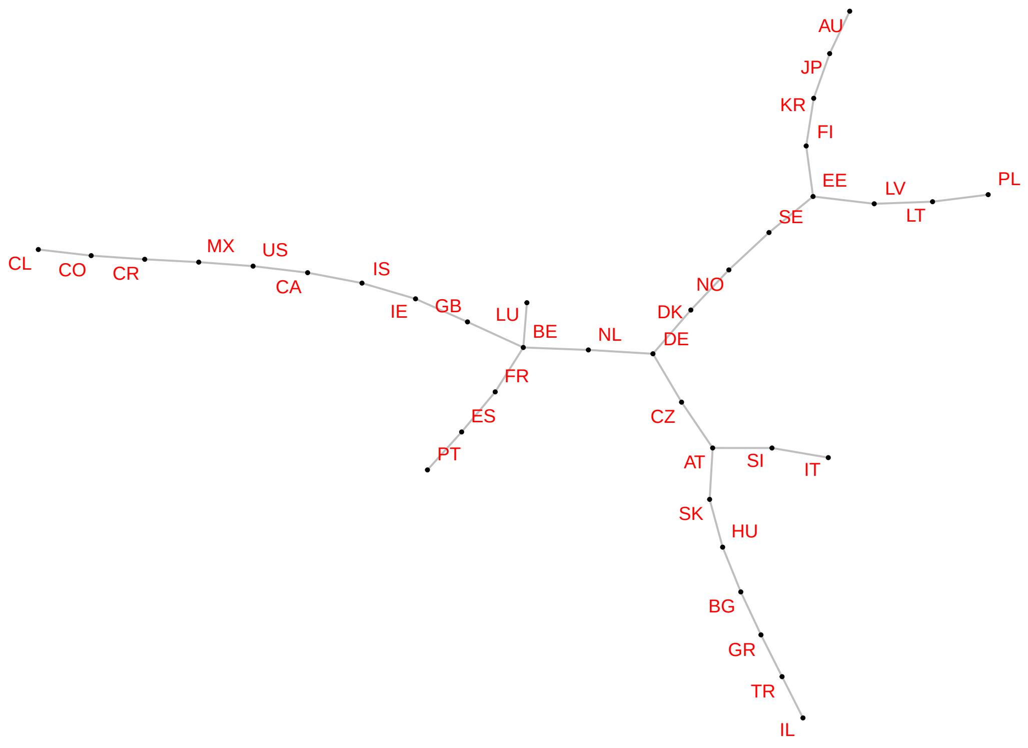

Figure 3.: Minimum Spanning Tree for Capital Cities of OECD Member States

3.4. Naive Heuristic Network

A key aim of this thesis is to explore alternative methods for constructing networks that do not rely on domain knowledge. To this end, we provide a simple heuristic-based network that acts as a benchmark for comparing the approaches discussed in this thesis. Our proposed knowledge-based network extends the MST described in Section 3.3 by connecting countries whose capital cities are within 600 kilometers of each other. The rationale behind this approach is that geographical proximity may act as a proxy for economic performance (Amidi and Majidi, 2020).

Figure 4 shows this proximity inferred network. The additional edges are coloured black, while the original MST edges remain grey. This process introduced 26 additional edges to the network. Hence, to ensure a fair comparison of performance, in later sections of this thesis we confine the network search space to networks with exactly 26 additional beyond the MST.

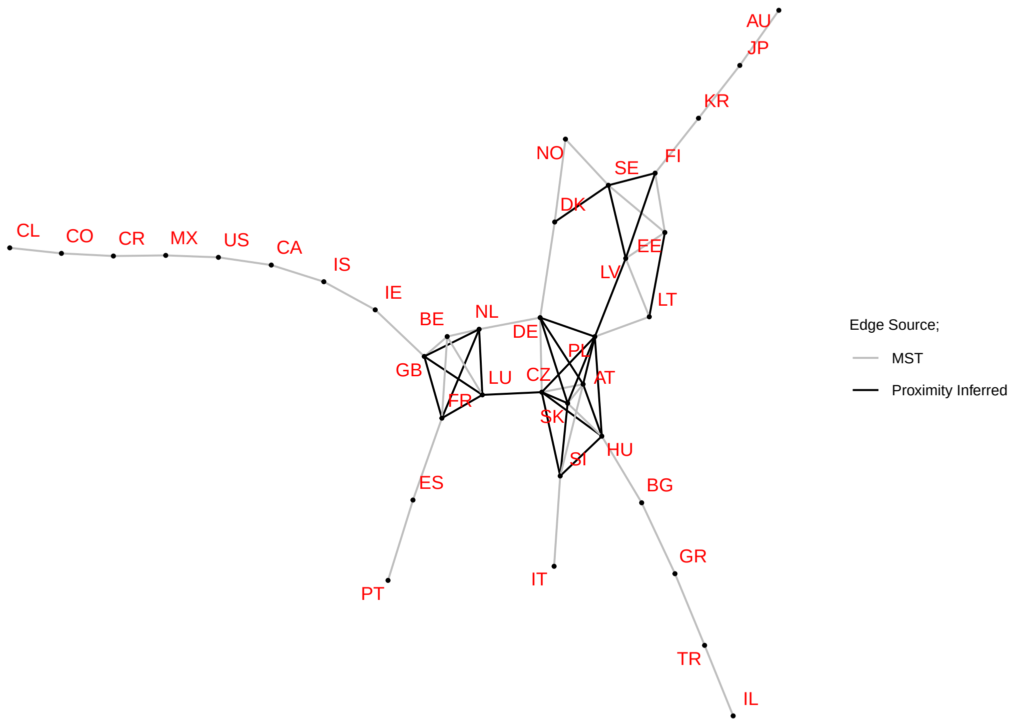

Figure 4.: MST with Proximity Inferred Edges

3.5. GNAR Selection

To guide the network searches in this thesis, we narrowed the focus of the search to a handful of model structures. In order to decide which models to include, we leveraged the valuable correlation-orbit (Corbit) plot, first introduced in Nason et al. (2023). Premised upon the well-known tool in time series analysis, the Autocorrelation Function (ACF), Corbit plots provide a visualisation for a network-analogous function known as the Network Autocorrelation Function (NACF).

The NACF, taking as input a network structure, a h-stage lag and a r-stage neighbour, calculates the autocorrelation for each time series $X_{i,t} \in \mathbf{X}_t$ by considering both the correlation between $X_{i,t}$ at time t and lagged version of itself $X_{i,t-h}$ as well as the influence of its r-stage neighbours and their lags. It then returns the ratio of the sum of these autocorrelations to the sum of the bounded version of these autocorrelations where this bound is determined by the constraints of the network. The result is a single value, quantifying the degree of autocorrelation present for a given network structure.

The Corbit plot provides an intuitive graphical aid for visualising the NACF values across various h-stage lags and r-stage neighbours simultaneously. More specifically, the plot allows us to study the autocorrelation decay, thus providing an indication of how we should configure our GNAR model for optimal performance. The rings of the Corbit plot shown in Figure 5 indicate the r-stage neighbours used to calculate the NACF. The innermost ring corresponds to 1-stage neighbours, the next ring towards the outside corresponds to 2-stage neighbours and so on. The number of steps anticlockwise from the 3 o'clock level reveals the depth of the model used, as indicated by the numbers 1 to 20 on the perimeter of the ring marking the depth levels.

The key idea here is to use the Corbit plot to identify an (h, r) pair where the NACF values are significantly different from zero. This indicates strong autocorrelation and tells us the lags and neighbour stages where the neighbouring nodes have are correlated with the current value and this may have a causal effect.

A noteworthy point is that the NACF doesn't account for the influence of previous lags in its calculation. This makes it difficult to discern model order using a Corbit plot with the NACF, since any observed auotcorrelation might not decrease due to the effects of earlier lags. The Partial Network Autocorrelation Function (PNACF) accounts for the influence of these lags and crucially removes them, aiding model order selection in the Corbit plot.

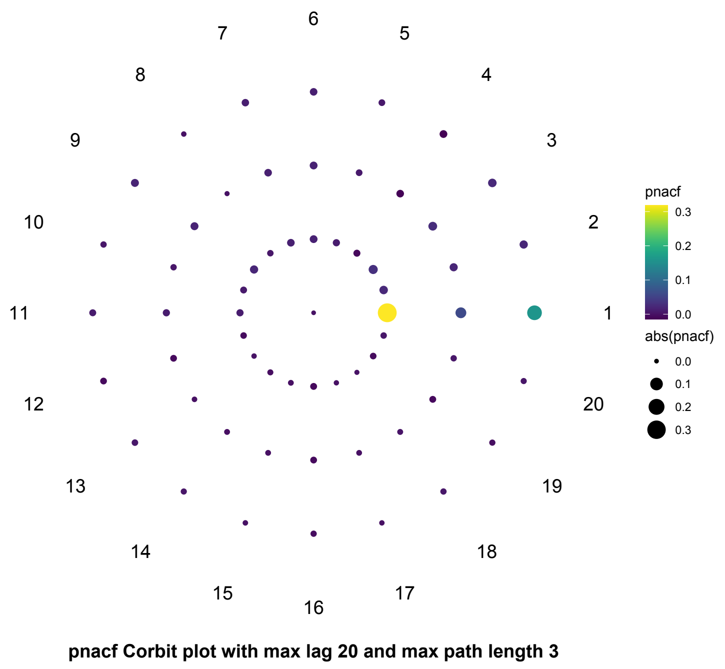

Figure 5.: Corbit plot for Proximity Inferred Input Network

Figure 5 shows the PNACF Corbit plot using the proximity-based heuristic network introduced in 3.4. The plot suggests that a GNAR model with a lag of 1 and a stage-neighbor component up to 3 is most appropriate for this dataset. Thus, in our network search we consider both global-α and local-α configurations of GNAR(1, [1]), GNAR(1, [2]) and GNAR(1, [3]) models.

4. Methods

4.1. Thompson Sampling

4.1.1. Multi Arm Bandit Problem

Multi Arm Bandit (MAB) problems, first formally introduced by Robbins (1952), describe scenarios where an agent faces a multitude of choices with no prior knowledge of the distribution of the reward achieved from the selection of any option or *arm*. The distribution of the reward achieved from each arm is assumed to be fixed and independent of the selection of previous arms. The agent has a fixed number of iterations of selection and their goal is to maximise the cumulative reward achieved over successive trials.

There is an inherent trade off in this problem between exploration and exploitation of rewards. Given the agent's objective of maximising cumulative reward, clearly the agent would rather exploit the arm that yields the highest average reward for as long as possible. On the other hand, the agent must explore the other arms to determine which of the choices yields the highest reward.

The network discovery problem at the heart of this thesis can be recast as a MAB problem wherein the arms are edges being added to an arbitrary network and the prediction accuracy associated with the addition of each edge is the reward. Crucially, the variability of the reward comes from the current graph to which the edge in question is being added to and also which additional edges preceded it. The reasoning for this is that there is assumed to be some sort of serial dependence amongst the addition of edges. Importantly, the degree of serial dependence here is unknown, should the edges have a low degree of serial dependence, we would expect the methodology mentioned in Section. 4.1.2 to perform well.

4.1.2. Thompson Sampling Algorithm

First introduced by Thompson (1933), Thompson Sampling (TS) algorithm has become a popular tool used in MAB problems. This Bayesian approach maintains a probability distribution over the expected rewards from each arm. Through iterative sampling and selection the algorithm updates theses distributions, elegantly balancing exploration of less certain options and exploitation known high-reward arms. This balance is struck by the algorithm selecting the option that is more likely to give a higher reward at each iteration.

The algorithm begins with specifying a non-informative prior distribution for the mean and variance of the reward for each arm. Following this, a random sample is drawn from each prior distribution, one sample per arm. The arm that produces the highest reward during this sampling phase is selected for one of the algorithm's iterations. The observed reward is recorded, and the posterior distribution of the reward for the chosen arm is updated using a Bayesian updating process.

This process of sampling and selecting arms repeats for a predefined number of iterations. The elegance of this approach lies in the use of uninformative priors over the reward distribution, combined with the updating procedure. This setup allows the algorithm to naturally balance exploration and exploitation. The high variance inherent in the uninformative priors increases the likelihood of sampling high rewards among previously untried options, thereby encouraging *exploration*.

Figure 6 illustrates how the distributions may evolve in the case where $n_{arms} = 3$. Here, the true means of the distributions are $1, 5$ and 10 for *Arm 1*, *Arm 2* and *Arm 3* respectively. The legend indicates the number of times each distribution has been sampled. We see in the top right panel of the figure that after 5 iterations, *Arm 3* has been sampled 3 times whereas *Arm 1* and *Arm 2* have only been sampled once. Two key observations are made; firstly, the posterior mean for *Arm 3* has converged closer to its true mean than the other arms and secondly, the variance of the distribution has decreased.

This is precisely how we expect the algorithm to operate. Over many iterations, we expect the algorithm to increasingly favour the higher-yielding arms versus the lower-reward arms. With selection the posterior means and variances for each arm is updated to more closely adhere to the sample means and variances.

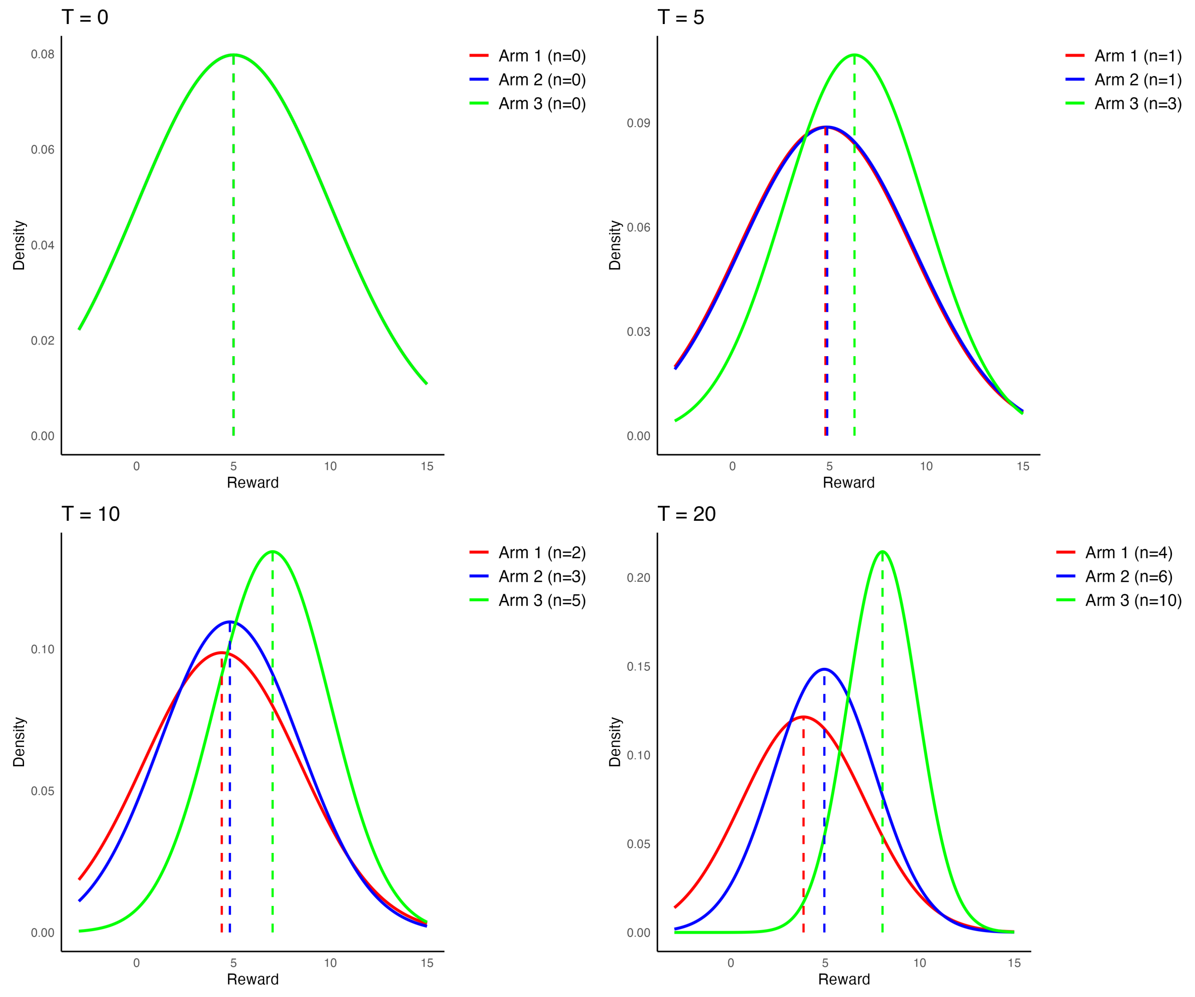

Figure 6.: Thompson Sampling Illustrative Example

TS has been applied in network optimisation problems in the past, particularly in the area of wireless communication to, improve network efficiency or *throughput*. For instance, Zhu et al. (2021) and Saxena et al. (2020) are two such papers, both introducing novel applications of TS to the network optimisation problem. However, both studies base their Bayesian update procedures on binary feedback based on success or failure of transmission. In contrast, our approach uses continuous feedback through the prediction error achieved when adding an edge to an arbitrary network.

TS has been applied in other graph theoretical settings in the past, notably the problem class known as *pathfinding* problems. In these scenarios, an agent must navigate through a network of nodes and edges to reach a destination while minimizing the journey's length. Importantly, the time it takes to traverse each edge is a random variable with an unknown distribution. Russo et al. (2020), describe how TS can be used to learn the distributions of the of the travel time associated with each edge.

Our network search problem, as detailed in Section 4.1.3, differs from these problems significantly. While we do explore distributions associated with edges, our objective is not to reach a predefined destination but to iteratively improve predictive performance. As far as we are aware this will be the first application of TS within the GNAR context.

4.1.3. GNAR Network Discovery as MAB Problem

At first glance, the relevance of multi-armed bandit problems to the network discovery problem central to this thesis may not be obvious. If we consider the negative prediction error (or reward) associated with adding an edge to a network, it is clearly evident that the reward is not independent of the current state of the network, pointing towards a sequential dependence amongst the edges.

However, rather than think of the addition of an edge to a specific network, in this thesis we attempt to find the distribution of the reward associated with the addition of an edge to an arbitrary or general network. The objective here is to find edges which are valuable in *most* cases. Once we have these distributions, we can select the edges that minimise the prediction error on average, thus constructing an optimised network.

A notable limitation of this approach is the assumption that edges with high individual rewards will also perform well when combined. If the aggregation of these edges results in high prediction errors, it suggests a significant degree of serial dependence.

4.1.4. Priors & Posterior Updates

We first define μ and σ^2 as the mean and variance of the prediction accuracy resulting from the addition of an edge to a general network. In our case, a *general network* refers to an arbitrary network that extends beyond the Minimum Spanning Tree (MST) by incorporating up to 26 additional edges, as discussed in Section 3.4.

We now make use the result from pg 25 Young and Smith (2005) to define a prior distribution over μ and σ^2. We assume the prior for σ^2 follows an Inverse-Gamma distribution with shape and scale parameters $\alpha > 0$ and $\beta > 0$. Further, we assume μ conditional on σ^2 follows a $N(v, \sigma^2/k)$ distribution for some $k > 0$ and $v \in \mathbb{R}$. Whence, the joint prior distribution of μ and σ^2 is Normal-Inverse-Gamma$(v, \kappa, \alpha, \beta)$. In summary;

$$\sigma^2 \sim \text{Inverse-Gamma}(\alpha, \beta) \tag{3}$$

$$\mu|\sigma^2 \sim \text{Normal}(v, \sigma^2/\kappa) \tag{4}$$

$$\mu, \sigma^2 \sim \text{Normal-Inverse-Gamma}(v, \kappa, \alpha, \beta) \tag{5}$$

Noting now that the prior distribution given by Equation 5 is in fact a conjugate prior, implying that the posterior distribution will be of the same parametric family. The parameters of the posterior distribution are as follows:

$$\alpha' = \alpha + \frac{n}{2}, \tag{6}$$

$$\beta' = \beta + \frac{1}{2}\frac{n\kappa}{n+\kappa}(\bar{x} - v)^2 + \frac{1}{2}\sum(x_i - \bar{x})^2, \tag{7}$$

$$\kappa' = \kappa + n, \tag{8}$$

$$v' = \frac{\kappa v + n\bar{x}}{\kappa + n} \tag{9}$$

A noteworthy point here is that the parameter κ, known as the *precision*, can be understood to control the confidence we have in the sample mean and sample variance over the initial priors. As shown by Equation 8, with every sample drawn, the confidence in the sample statistics should grow with more samples. The impact of this precision parameter is noted in Equation 7 and Equation 9 where the respective sample statistics are lent more weight with an increasing number of samples.

4.1.5. Thompsons Network Discovery Algorithm

To recast the GNAR network discovery problem within the context of Multi-Armed Bandit (MAB) frameworks, some adaptations from traditional approaches are necessary, as discussed in Section 4.1.3.

In this thesis, each iteration involves constructing a network with 26 edges beyond the Minimum Spanning Tree (MST). During each iteration, we sample and select a unique arm 26 times. After each edge addition, we update the posterior distribution of each arm and, importantly, store the updated graph. This process enables the construction of a new network at each step. Hence, the posterior distributions generated will reflect the reward distribution associated with adding an edge an arbitrary network. Once the algorithm terminates, we then select the k arms with the highest mean rewards (negative prediction error) to construct a network of edges that perform the best on average. Psuedo-code to describe this process is provided in Algorithm 1.

Algorithm 1 Thompson Sampling with Normal-Inverse-Gamma Prior for Additional Edge Network Construction

```
1: Input: Set of available arms $\mathcal{I}$, number of iterations $\mathcal{T}$, target number of additional edges $\mathcal{M}$, Base graph $\mathcal{G} = (\mathcal{K}, \mathcal{E})$
2: Initialise:
3:     For each $i = 1 \in \mathcal{I}$:
4:         Normal-Inverse-Gamma prior parameters: $v_i$, $\kappa_i$, $\alpha_i$, $\beta_i$
5:         Initial counts: $n_i = 0$
6:
7: For $t = 1$ to $\mathcal{T}$:
8:   Initialise:
9:   Set to store chosen arms: $\mathcal{C} = \varnothing$
10:  Base graph: $\mathcal{G}' = \mathcal{G}$
11:
12:  For edge in $m = 1, \dots, \mathcal{M}$:
13:    Initialise:
14:    Set to store posterior samples: $P = \varnothing$
15:
16:    For each arm $i = 1, \dots, \mathcal{I}$:
17:      Sample $\sigma_i^2$ from posterior distribution: $\sigma_i^2 \sim IG(\alpha_i, \beta_i)$
18:      Sample $\mu_i$ from the posterior distribution: $\mu_i \sim N(v_i, \sigma_i^2/\kappa_i)$
19:      Update set of posterior samples: $P = P \cup \{\mu_i\}$
20:    End For
21:
22:  Choose Arm:
23:    Find $i^*$ such that $max(P) = \mu_{i^*}$ for $i^* \in \mathcal{K} \setminus \mathcal{C}$
24:    Update chosen arm set: $\mathcal{C} = \mathcal{C} \cup \{i^*\}$
25:
26:  Observe reward:
27:    Add arm $i^*$ to graph: $\mathcal{G} = (\mathcal{K}, \mathcal{E} \cup \{i^*\})$
28:    Fit GNAR on $\mathcal{G}$ & store negative prediction error as reward $r_{i^*}$
29:
30:  Perform Bayesian Update:
31:    $n_{i^*} \leftarrow n_{i^*} + 1$
32:    $\bar{r}_{i^*} \leftarrow \frac{\bar{r}_{i^*}(n_{i^*}-1)+r_{i^*}}{n_{i^*}}$
33:    $v_{i^*} \leftarrow \frac{\kappa_{i^*}v_{i^*}+r_{i^*}}{\kappa_{i^*}+1}$
34:    $\kappa_{i^*} \leftarrow \kappa_{i^*} + 1$
35:    $\beta_{i^*} \leftarrow \beta_{i^*} + \frac{1}{2}\frac{\kappa_{i^*}(\bar{r}_{i^*}-v_{i_*})^2}{\kappa_{i^*}+1} + \frac{1}{2}\sum(r_{i^*} - \bar{r}_{i^*})^2$
36:    $\alpha_{i^*} \leftarrow \alpha_{i^*} + \frac{1}{2}$
37:  End For
38: End For
39: Return: Posterior distribution of reward associated with each edge
```

4.2. Genetic Algorithm

4.2.1. Genetic Algorithm Overview

First introduced by Holland (1975) and inspired by Darwinian evolutionary theory, the Genetic Algorithm (GA) is a population-based optimization algorithm. GA is premised upon on the principle of *survival of the fittest*, where, in each iteration, a population of candidate solutions undergoes genetic operations—such as selection, crossover, and mutation to generate a new population of solutions. Each solution's ability to solve the task at hand, or it's *fitness*, is evaluated using a fitness function, which in our case is GNAR within-sample predictive accuracy. This fitness influences the likelihood of a solution being selected to reproduce, with fitter solutions being more likely to contribute to the next generation.

As highlighted by Kemp (2006), GAs offer several advantages over traditional search methods. First, GAs do not require the calculation of gradients, reducing computational demands. Second, GAs are population-based rather than single-point-based, making them suitable for exploring large solution spaces and reducing the risk of getting trapped in local optima. Finally, GAs use probabilistic decision-making instead of the deterministic approaches typical of gradient-based search algorithms, resulting in more robust search processes.

GAs have been used for a wide variety of problems in the past, some fields include; optimisation of wireless sensor networks (Norouzi and Zaim, 2014), biological networks (Kikuchi et al., 2003), facility layout (Datta et al., 2011) and information security (Kaur and Kumar, 2018).

A prominent use of GAs in the literature has been in the optimisation of neural networks (Chiroma et al., 2017; Ding et al., 2011; Xie and Yuille, 2017). One paper of particular interest is Xie and Yuille (2017), wherein the authors used GAs to search for optimal Convolutional Neural Network (CNN) structures, in the paper they limit the number of stages permitted to be use in the model architecture. Further, they introduce a novel method for encoding the CNN structure. This encoding and limiting of stages allowed the authors to perform a fixed-length binary GA search. For each architecture generated, they train the model and evaluate its performance versus reference data. This process is similar to that which we implement in this thesis, except rather than searching for neural network architectures, we encode network configurations and instead of testing versus reference data we evaluate performance using within-sample prediction accuracy. Further, we introduce novel custom genetic operations to allow for an efficient constrained network search as outlined in Section 4.2.4.

Noting that a *chromosome* in the context of GAs refers to solution encoded as a 1d-vector. In this thesis, we work only with binary GAs, hence are only concerned with solutions encoded as binary vectors as shown in Fig. 7.

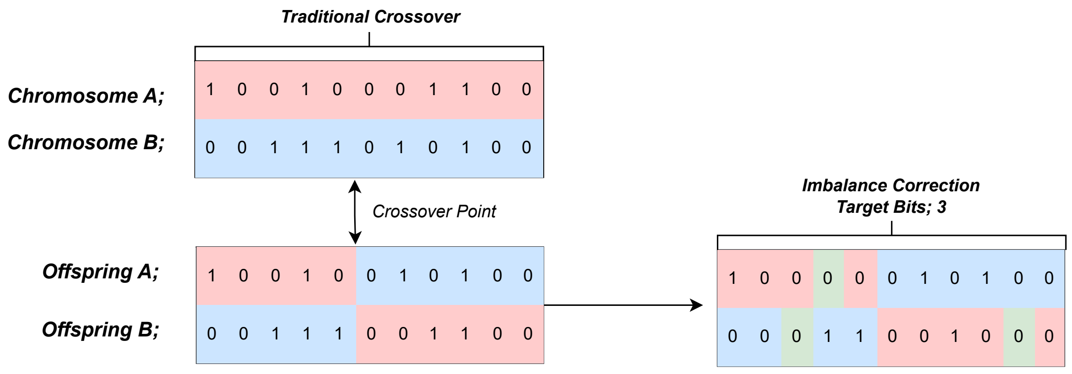

Figure 7.: Crossover Imbalance Correction

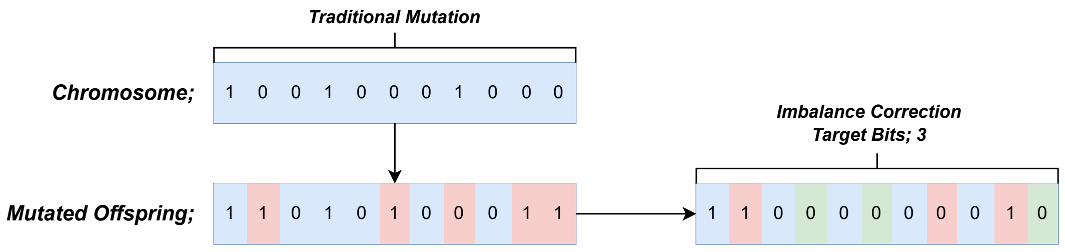

Figure 8.: Mutation Imbalance Correction

For a graph $\mathcal{G}$ with only undirected edges and no self-loops, its adjacency matrix $\mathcal{A}$ can be fully described by the upper triangular portion of $\mathcal{A}$. This is because undirected edges imply bidirectional connections, making $\mathcal{A}$ symmetric. As a result, the flattened upper triangular section of adjacency matrices is particularly well-suited for our network search. It's important to note that while these methods can be adapted for graphs with directed edges, doing so would naturally entail increased computational complexity.

4.2.2. Traditional Genetic Operations

The **Crossover** operation takes as input, two *parent* chromosomes and exchanges a portion of their genes with each other beyond a certain, randomly chosen point known as the *crossover point*. The standard crossover operation is shown on the left-hand side of Figure 7. The **Crossover Probability**, p_C governs the frequency at which two selected parents crossover.

Occurring, after the crossover operation takes place, the **Mutation** operation takes a chromosome as input and randomly switches a random selection of bits. The goal here is to introduce diversity into the population and avoid convergence to a locally optimum solution. The **Mutation Probability**, p_M, controls the frequency at which individual bits are flipped within a chromosome.

Selection is a key step in Genetic Algorithms, where the algorithm determines which chromosomes will be chosen to participate in the crossover process to produce new off-spring. This process directly impacts the algorithm's ability to evolve and improve solutions over time. Extensive research has explored various selection methods to optimise this process. These methods differ in how they prioritise chromosomes based on their fitness and diversity. For a comprehensive overview of these approaches, see Shukla et al. (2015), which provides a detailed comparative review of different selection strategies.

In this thesis, we make use of a combination of two selection strategies; **Roulette Wheel Selection** and **Elitism**. In *Roulette Wheel Selection*, chromosomes are randomly selected to be parents, where the probability of selection is proportional to their fitness. This ensures that the fittest members of the population have the highest chance of producing offspring whilst still maintaining some diversity. *Elitism* is a policy, first introduced by DeJong (1975), which ensures that a select few of the most fit chromosomes in the population are guaranteed to be selected as parents.

4.2.3. Additional Edges & Fitness Function

Applying genetic algorithms to the problem of finding a network with exactly 26 additional edges beyond the MST, requires some adaptation from traditional approaches.

In our case, with 37 nodes representing OECD member states, there are a total of 666 undirected edges possible in the network. However, since this thesis focuses on extending the Minimum Spanning Tree (MST) - which consists of 36 edges - the number of edges in consideration is in fact 630. Finally, we restrict our search to networks that include 26 edges beyond the MST, as outlined in the heuristic-based approach introduced in Section 3.4. Consequently, the size of our solution space is given by $\binom{630}{26} \approx 8.914 \times 10^{45}$, representing the number of possible networks with 26 edges added to the MST.

With these considerations in mind, it was essential introduce the custom crossover, mutation and initialisation functions outlined in Section 4.2.4 to coerce the algorithm into finding eligible solutions efficiently. Otherwise, the likelihood of the algorithm successfully manipulating a binary vector of 630 to produce exactly 26 additional edges would be extremely slim.

Lastly, a key technical aspect of the search algorithm is that it operates by identifying and modifying indices within the flattened upper triangular matrix of the MST. Specifically, the algorithm seeks out *zero bins*—entries that are currently set to zero—and flips them to one in order to construct an adjacency matrix containing 26 additional edges to the MST.

4.2.4. Custom Operations

Custom Crossover Operation

Using the standard crossover as defined in Section 4.2.2 can lead to offspring with a number of additional edges differing from our target 26. To remedy this, we introduce a custom crossover operation. After the usual crossover operation takes place, if the resulting offspring have a number of additional edges that differs from our target, we randomly select the appropriate number of bits to flip in order to yield offspring with the exact number of target additional edges required. An illustration of this custom crossover process is provided in Figure 7. Similarly, we introduce a custom mutation operation that corrects for this edge imbalance following the standard mutation operation. Figure 8 provides a visualisation for this process. These modifications allow us to focus our search exclusively on viable solutions while retaining the majority of the genetic characteristics inherited from the parent solutions.

Finally, Figure 9 presents an illustrative flowchart that outlines the working mechanism of our GA. Additionally, Algorithm 2 offers pseudo-code detailing the network search process.

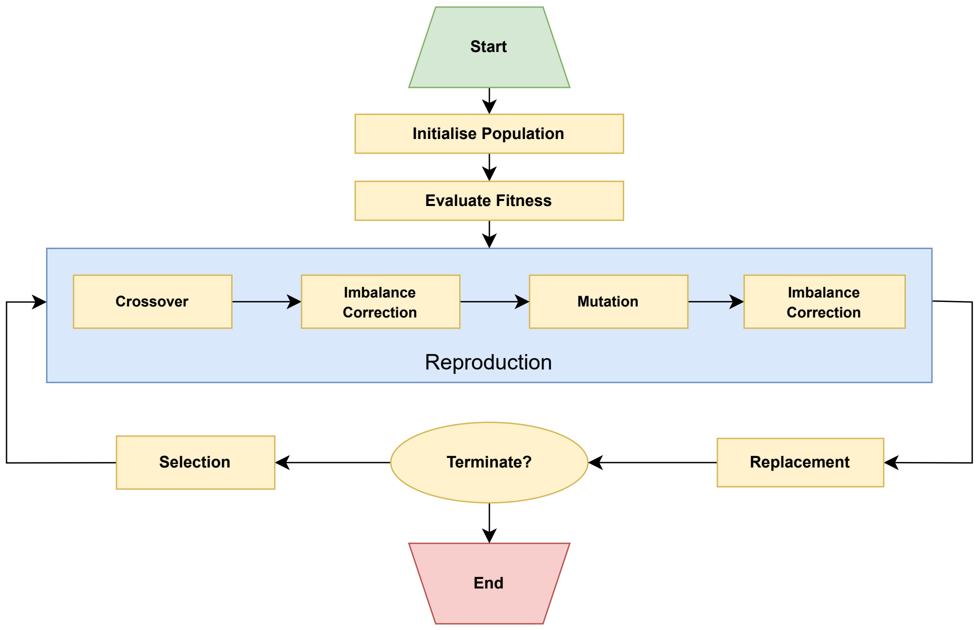

Figure 9.: Genetic Algorithm Schematic

Algorithm 2 Genetic Algorithm for GNAR Network Search

```
1: Input: Population size $N$, number of generations $\mathcal{G}$, crossover probability $p_c$, mutation probability $p_m$, fitness function $f$
2: Custom Initialisation:
3:     Generate population $\mathcal{P}_0$ of size $N$ of networks exactly 26 edges beyond MST
4:
5: For $g = 1$ to $\mathcal{G}$:
6:   Evaluate fitness (GNAR Predictive Accuracy):
7:     Calculate fitness for each individual in population: $f_i = f(\mathcal{P}_g[i])$ for $i = 1, \ldots, N$
8:
9:   Selection:
10:    Using Roulette Wheel & Elitism select chromosomes to form a mating pool
11:
12:  Custom Crossover:
13:    For each pair of parents in the mating pool:
14:        With probability $p_c$, perform crossover to produce offspring
15:        Correct edge imbalance
16:        Add offspring to the new population $\mathcal{P}_{g+1}$
17:    End For
18:
19:  Custom Mutation:
20:    For each individual in $\mathcal{P}_{g+1}$:
21:        With probability $p_m$, mutate the individual
22:        Correct edge imbalance
23:    End For
24:
25:  Replacement:
26:    Replace the old population $\mathcal{P}_g$ with the new population $\mathcal{P}_{g+1}$
27:
28: End For
29: Return: The best individual found across all generations
```

5. Results

To evaluate the performance of the methods introduced in this thesis, we applied each approach to a variety of GNAR models with different specifications. Namely, we investigated GNAR models with one lag and up to three neighbours, fitting both global and local-α configurations in each case. On top of this, for each configuration, we generated 10^4 random networks and stored the prediction error for the best performing network - matching the number of solutions generated by the GA. These random networks were designed to provide a brute-force comparison to our novel approaches. The resulting prediction errors for each approach are presented in Table 1.

Thompson Sampling

To implement the Thompson sampling approach, we allowed for 1000 iterations of sampling and selecting arms, with each iteration corresponding to the sequential addition of 26 edges to the network. The initial parameters for each arm were set as follows; $v_0 = 0, \kappa_0 = 1, \alpha_0 = 1$ and $\beta_0 = 1$. Figure. 10 shows the training history for each of the searches, a notable takeaway from these plots is the clear exploratory period seen in the searches using the GNAR$(1, [2])$ and GNAR$(1, [3])$ global and local-α GNAR specifications.

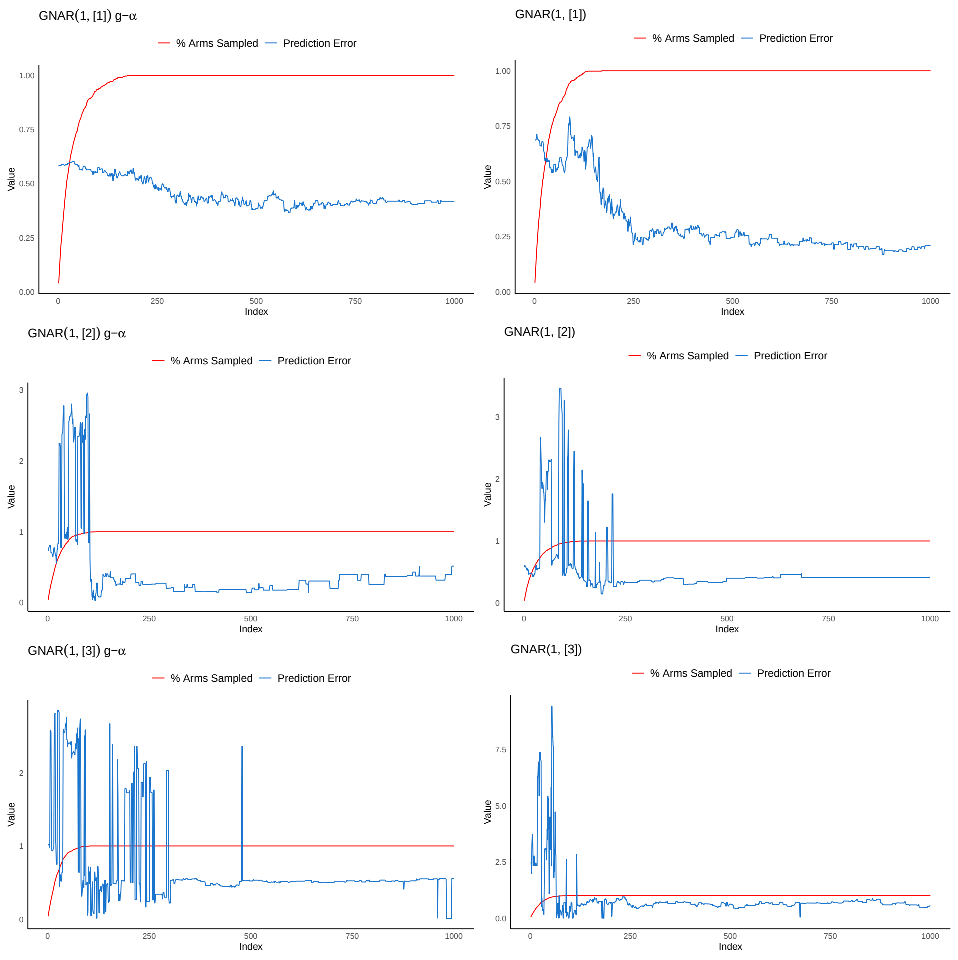

Figure 10.: Thompson Sampling training history, here 'g - α' refers to a global - α GNAR model

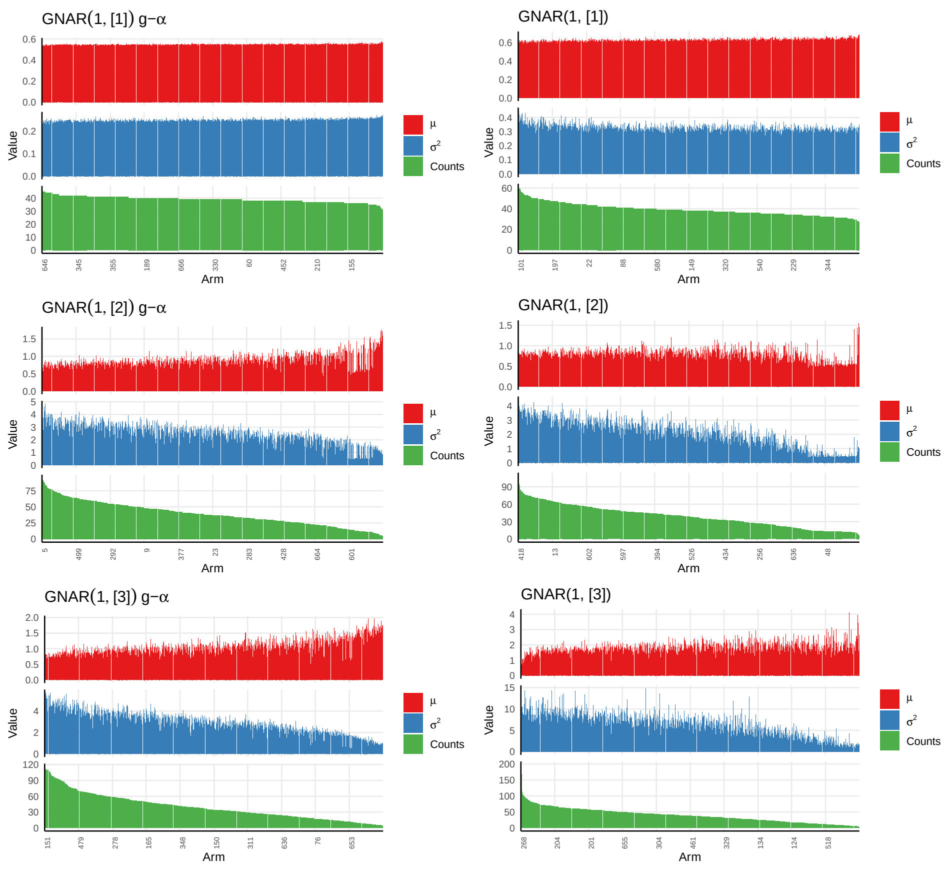

Figure 11.: Thompson Sampling bar chart showing μ, σ^2, and the number of times sampled for each arm, ordered from left to right by the most frequently sampled arms.

Genetic Algorithm

In our Genetic Algorithm searches, we consider a population size of 100 at each iteration. We set the crossover probability $p_C = 0.8$ and the mutation probability, $p_M = 0.1$. We also consider five solutions to be *elite* in every population i.e. in each population the fittest five solutions were guaranteed to be parents of subsequent offspring. The training history for each GA search is shown in Figure 12. We note that for the global and local-α GNAR(1, [3]) configurations, the search arrives at a near zero prediction error after the first iteration.

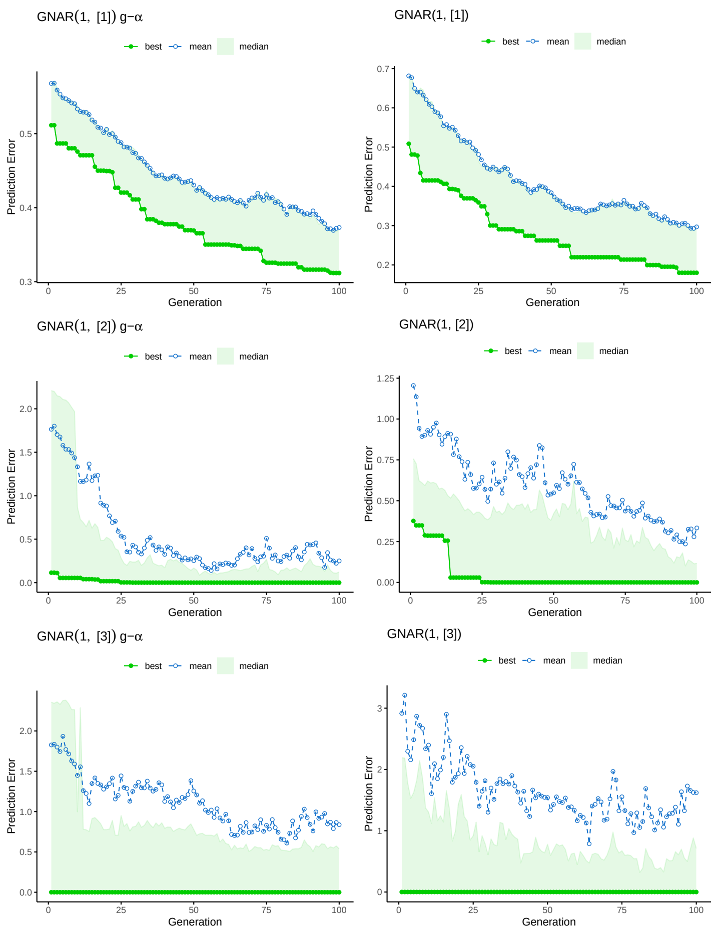

Figure 12.: Genetic Algorithm training history, here ‘g - α’ refers to a global - α GNAR model

Table 1.: Prediction Error for Various Networks w/ Lag, $p = 1$

	Global Alpha			**Local Alpha**		
Network	$s = [\mathbf{1}]$	$s = [\mathbf{2}]$	$s = [\mathbf{3}]$	$s = [\mathbf{1}]$	$s = [\mathbf{2}]$	$s = [\mathbf{3}]$
MST	0.592	0.645	0.945	0.683	0.634	2.43
Heuristic	0.595	0.586	0.695	0.704	0.755	1.41
Random	0.464	0.0953	1.77×10^{-6}	0.389	0.004	2.2×10^{-7}
Thompson	0.419	0.515	0.554	0.211	0.413	0.561
Genetic	0.312	2.94×10^{-8}	7.72×10^{-8}	0.18	2.39×10^{-9}	7.18×10^{-8}

6. Discussion

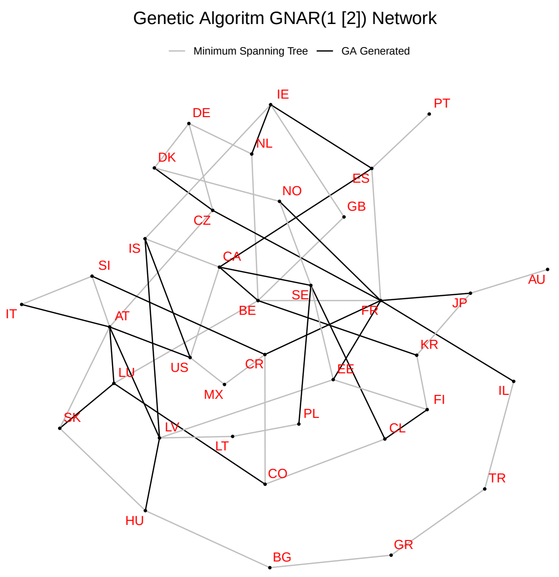

Figure 13.: Genetic Algorithm GNAR(1, [2]) Generated Network

Taking a closer look at the performance of each method shown in Table 1, the most striking result is the outstanding performance of the genetic algorithm, which outshines all other methods. With near-zero prediction errors across each model specifications, the genetic algorithm is clearly shown to be the superior approach. However, As highlighted in Section 2.0.3, it is important to note that the average monthly difference in unemployment rates across all included countries is quite small (≈ -0.01). Thus, given the GNAR model's autoregressive nature, the near zero prediction errors appear less surprising. Figure 13 shows the network generated by the best performing genetic algorithm, namely the GNAR(1, [2]) model. The resulting network gave France an additional six edges over the MST, this somewhat aligns with expectations when we recall that France's status as the third largest economy in Europe (International Monetary Fund, 2024). With regards to the other connections, many lack an obvious economic connection. We remind the reader that the solution space for this problem is extremely large and that a longer and more in depth search could yield more intuitive results.

Despite the TS approach sampling each arm at least three times for each configuration as shown in Figure 11, it failed to outperform the brute force approach in nearly every case. This points towards a serial dependence in the addition of edges. In other words, since the resulting network is constructed from the best performing edges on average when added to an arbitrary network, since these edges do not perform well cumulatively, this indicates that there is a dependence amongst the edges.

Another noteworthy observation is the sensitivity of the prediction error to the underlying input network as model complexity increases. This is clearly demonstrated by the volatile swings during the exploratory phase of Thompson sampling, as shown in Figure 10. This pattern is supported in the GA's training history, where both the mean and median prediction errors are notably distant from the optimal solutions within the population. This suggests that the model's performance is more heavily influenced by the input network structure for more complex models, stressing the importance of a carefully considered input network structure.

In conclusion, we firstly note that both novel approaches consistently outperform the naive proximity-inferred network, leading to significant improvements in forecasting accuracy across all cases - indicating a successful search in this regard. Further, through this research we provide detailed comparison network search using a Bayesian and meta-heuristic search algorithm in a GNAR centred problem setting. That said, the TS search failed to outperform the brute force approach in most cases, indicating a level of serial dependence amongst the edges of the network. On the other hand, the GA produced excellent results, beating all comparison models in every case. The training history for the both novel approaches drew attention to the volatile swings in prediction error with different input network, emphasising the importance of an objective network search. Another major contribution of this thesis is the proposed methodology for identifying comparable networks to those constructed using domain knowledge. This approach can not only validate economic theory but also has the potential to uncover anomalous results that may yield valuable real-world insights.

7. Further Research

A sensible start point for future research would be to relax some of the restrictions of the methodology proposed in this thesis. One such area would be in model selection; recalling that in Section 3.5 we used the Corbit plot to provide an indication of what range of GNAR specifications to use. However, testing many GNAR models, as we have, can be computationally expensive. On top of this, the model parameterisation indicated the Corbit plot may not be consistent for newly generated networks. It is possible that an optimised model configuration exists for networks created using the proposed methodology, which could lead to improved performance. A search algorithm making use of the NACF and PNACF discussed in 3.5 running in tandem with a network search could lead to further improvements in predictive accuracy.

Another direction for future research could involve incorporating a residual analysis into the proposed algorithm. Penalising networks with high residual levels may lead to more robust solutions. Given that the **GNARfit** function in the **GNAR** package is built on the widely used **R** function **lm**, it would be relatively straightforward to leverage the residual analysis tools already available in **R** for this purpose.

References

Amidi, S. and Majidi, A. F. (2020). Geographic proximity, trade and economic growth: a spatial econometrics approach. *Annals of GIS*, 26(1):49–63.

Arnold, A., Liu, Y., and Abe, N. (2007). In *Temporal causal modeling with graphical granger methods*, KDD '07, page 66–75, New York, NY, USA. Association for Computing Machinery.

Chan-Lau, J. (2017). Variance decomposition networks: Potential pitfalls and a simple solution. *IMF Working Papers*, 17:1.

Chiroma, H., Noor, A. S. M., Abdulkareem, S., Abubakar, A. I., Hermawan, A., Qin, H., Hamza, M. F., and Herawan, T. (2017). Neural networks optimization through genetic algorithm searches: a review. *Appl. Math. Inf. Sci*, 11(6):1543–1564.

Datta, D., Amaral, A., and Figueira, J. (2011). Single row facility layout problem using a permutation-based genetic algorithm. *European Journal of Operational Research*, 213:388–394, http://dx.doi.org/10.1016/j.ejor.2011.03.034.

DeJong, K. (1975). *An Analysis of the Behavior of a Class of Genetic Adaptive Systems.* Ph.d. dissertation, University of Michigan, Ann Arbor.

Diebold, F. X. and Yılmaz, K. (2014). On the network topology of variance decompositions: Measuring the connectedness of financial firms. *Journal of Econometrics*, 182(1):119–134. Causality, Prediction, and Specification Analysis: Recent Advances and Future Directions.

Ding, S., Su, C., and Yu, J. (2011). An optimizing BP neural network algorithm based on genetic algorithm. *Artif. Intell. Rev.*, 36:153–162.

Friedman, J., Hastie, T., and Tibshirani, R. (2007). Sparse inverse covariance estimation with the lasso.

Granger, C. W. J. (1969). Investigating causal relations by econometric models and cross-spectral methods. *Econometrica*, 37(3):424–438.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, 2nd edition. 2nd Edition, MIT Press, 1992.

International Monetary Fund (2024). World economic outlook database: April 2024 edition. https://www.imf.org/en/Publications/WEO/weo-database/2024/April.

Accessed: 2024-08-24.

Kaur, M. and Kumar, V. (2018). Beta chaotic map based image encryption using genetic algorithm. *International Journal of Bifurcation and Chaos*, 28(11):1850132.

Kemp, R. (2006). An introduction to genetic algorithms for neural networks. *University of Cambridge, UK*, 5.

Kikuchi, S., Tominaga, D., Arita, M., Takahashi, K., and Tomita, M. (2003). Dynamic modeling of genetic networks using genetic algorithm and S-system. *Bioinformatics*, 19(5):643–650.

Knight, M., Leeming, K., Nason, G., and Nunes, M. (2019). Generalised network autoregressive processes and the GNAR package.

Knight, M. I., Nunes, M. A., and Nason, G. P. (2016). Modelling, detrending and decorrelation of network time series.

Mantziou, A., Cucuringu, M., Meirinhos, V., and Reinert, G. (2023). The GNAR-edge model: A network autoregressive model for networks with time-varying edge weights.

Nason, G., Salnikov, D., and Cortina-Borja, M. (2023). New tools for network time series with an application to covid-19 hospitalisations.

Nason, G., Salnikov, D., and Cortina-Borja, M. (2024). Modelling clusters in network time series with an application to presidential elections in the usa.

Nason, G. P. and Wei, J. L. (2022). Quantifying the Economic Response to COVID-19 Mitigations and Death Rates Via Forecasting Purchasing Managers' Indices Using Generalised Network Autoregressive Models with Exogenous Variables. *Journal of the Royal Statistical Society Series A: Statistics in Society*, 185(4):1778–1792.

Norouzi, A. and Zaim, A. H. (2014). Genetic algorithm application in optimization of wireless sensor networks. *The Scientific World Journal*, 2014(1):286575.

OECD (2024). Oecd data explorer. Accessed: 2024-08-22.

Robbins, H. (1952). Some aspects of the sequential design of experiments.

Russo, D., Roy, B. V., Kazerouni, A., Osband, I., and Wen, Z. (2020). A tutorial on thompson sampling.

Saxena, V., Gonzalez, J. E., Stoica, I., Tullberg, H., and Jaldén, J. (2020). Constrained thompson sampling for wireless link optimization.

Scrucca, L. (2013). GA: A package for genetic algorithms in R. *Journal of Statistical*

Software, 53(4):1–37.

Shukla, A., Pandey, H. M., and Mehrotra, D. (2015). Comparative review of selection techniques in genetic algorithm. In *2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE)*, pages 515–519.

Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294.

Xie, L. and Yuille, A. (2017). Genetic cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1388–1397.

Young, G. and Smith, R. (2005). *Essentials of Statistical Inference*. Cambridge University Press.

Zhu, J., Huang, X., Gao, X., Shao, Z., and Yang, Y. (2021). Multi-interface channel allocation in fog computing systems using thompson sampling. *IEEE Internet of Things Journal*, 8(17):13542–13554.

Zhu, X., Wang, W., Wang, H., and Härdle, W. K. (2019). Network quantile autoregression. *Journal of Econometrics*, 212(1):345–358.

A. Alpha-2 ISO Code Lookup

Flag	**ISO Code**	**Country**	**Flag**	**ISO Code**	**Country**
🇦🇺	AU	Australia	🇯🇵	JP	Japan
🇦🇹	AT	Austria	🇰🇷	KR	South Korea
🇧🇪	BE	Belgium	🇱🇻	LV	Latvia
🇧🇬	BG	Bulgaria	🇱🇹	LT	Lithuania
🇨🇦	CA	Canada	🇱🇺	LU	Luxembourg
🇨🇱	CL	Chile	🇲🇽	MX	Mexico
🇨🇴	CO	Colombia	🇳🇱	NL	Netherlands
🇨🇷	CR	Costa Rica	🇳🇴	NO	Norway
🇨🇿	CZ	Czech Republic	🇵🇱	PL	Poland
🇩🇰	DK	Denmark	🇵🇹	PT	Portugal
🇪🇪	EE	Estonia	🇸🇰	SK	Slovakia
🇫🇮	FI	Finland	🇸🇮	SI	Slovenia
🇫🇷	FR	France	🇪🇸	ES	Spain
🇩🇪	DE	Germany	🇸🇪	SE	Sweden
🇬🇷	GR	Greece	🇹🇷	TR	Turkey
🇭🇺	HU	Hungary	🇬🇧	GB	United Kingdom
🇮🇸	IS	Iceland	🇺🇸	US	United States
🇮🇪	IE	Ireland			
🇮🇱	IL	Israel			
🇮🇹	IT	Italy			