UNIVERSITY OF COPENHAGEN

Niels Bohr Institute

MASTERS THESIS, 45 CREDIT HOURS

Structural properties and expression pattern of evolved Boolean networks

Author:
Alisa AGAFONOVA

Supervisor:
Dr. Kim SNEPPEN

Biocomplexity and Biophysics Group
Niels Bohr Institute

March 30, 2021

Abstract

We conducted a numerical study of evolution in structure and expression pattern of Boolean networks, using the idea that the expression pattern is step-wise preserved during the evolutionary process as proposed by Bornholdt and Sneppen in 1998 [Bornholdt and Sneppen, 1998], with the addition of logical signal tracing along all three-node long paths. The evolved networks exhibited stable structural properties and degree distributions wider than the expected Poisson distribution for random networks. The evolved networks with higher average link density had large frozen-ON components, not observed in randomized networks.

Acknowledgements

I would like to thank my master's thesis supervisor, Kim Sneppen for his kindness, Olympian patience and for all the science. It was a lot of fun! Numerous other people helped me to complete my masters thesis. I would like express my gratefulness for their help, and especially thank Bente Markussen and Namiko Mitarai.

Contents

List of Figures

Chapter 1

Introduction

1.1 Boolean logic and biological networks

Biological regulation are largely logical, with transcription factors that either activate or repress each others expression. An example is given below, showing the transcription regulatory network of phage lambda reproduced from [Trusina A, 2005] with positive and negative regulations highlighted in green and red arrows, respectively. The effective regulation is defined not only by the logic of the individual input, *i.e.* whether the input is a repressor or an activator, but also by the way this logic is organized. For example, then A and B is different than A or B, as regulatory input to a third transcription factor C.

Following Kauffman we restrict ourselves to nested canalizing Boolean combinations meaning that we for example do not allow the exclusive or. Canalizing regulation is one where there at least one of the inputs overrides all other inputs when it takes a specific value, for example +1. A nested canalizing input utilizes this feature repeatedly in a hierarchical ordering. In this project we will evolve Boolean networks with nested canalization, as models of transcription factor networks.

We focus on transcription factors only, and therefore restrict the model networks to nodes that can have regulatory input only. In doing so we disregard the vast majority of proteins in the cell which do not participate in regulation, and have role in structure-related or metabolic function. In the phage network below, the cyan nodes are the transcription factor network of the phage, and the yellow are the signal input node from the bacteria (that sometimes work by degradation).

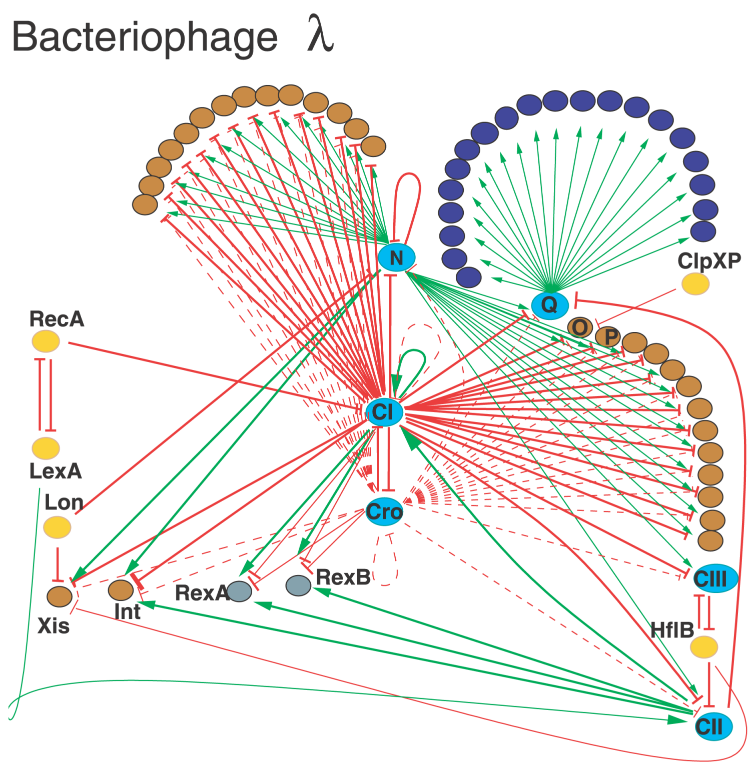

FIGURE 1.1: Genetic Regulatory Network of Phage λ. The proteins are colored according to their function and expression mode, with cyan nodes representing transcription factors. The positive regulation is shown with green arrows, and negative regulation (*i.e.* repression) is marked with red arrow. Reproduced from [Trusina A, 2005]

1.2 Signal paths

Suppose the transcription factor CII of the λ bacteriophage Genetic Regulatory Network on Figure 1.1 sent an 'activation signal' to CI, and CI, in return, acted as repressor on all five remaining transcription factors. In a larger network, with more branching and path variation, how far does regulatory 'signal' from a transcription factors spread? In this model the effect off each new stimuli is traced along all three-node paths, as longer paths would be too time-consuming in nature, and too perturbed by alternative signaling to be relevant.

Chapter 2

Methods

2.1 Evolution model

This project is a numerical study of evolution in structure and expression patterns of Boolean networks, using the idea that the expression pattern is step-wise preserved during the evolutionary process as proposed by Bornholdt and Sneppen in 1998. [Bornholdt and Sneppen, 1998], with the addition of logical signal tracing along all three-node long paths.

We used directed Boolean networks as model for the regulatory interactions between transcription factors in the Gene Regulatory Network. In a fully connected network, one can follow the edges (regulatory connections) from one node to any other node. Of course the biological interactions between transcription factors are not always directly connected and often are not indirectly connected. In this model, the effect of each new stimuli (a transcription factor expression state change, or a new outside condition, such as temperature shift) is traced over all three-node long paths. The effect of longer tracing was found negligible in simulation, i.e. the result would not be different if we for example considered 5 steps path. In real cells one would imagine that longer paths may be too time consuming, and too perturbed by alternative signaling to be relevant.

To reflect the destructive potential of each new mutation, regulatory connections the node receives are prioritized based on their evolutionary age. Thus nodes receiving edges from more than one node are assigned nested Boolean functions ranking the inputs in the order they were first introduced, with the oldest mutation first, and the current/newest mutation last.

In each mutation cycle, the expression pattern is determined by performing simultaneous update along all three-node-long interaction paths for both the original network and for the mutant. In the event that mutant expression pattern over the entire cycle equals that of the original network, the mutation is deemed non-destructive, and is accepted.

Link-rewiring mutations as well as node duplication and node loss are implemented. As the model is restricted to transcription factors only, nodes that do not have any regulatory connections are removed at the end of each mutation cycle.

2.2 Mutation cycle

The mutation cycle generally follows the rules outlined in [Bornholdt and Sneppen, 1998] adjusted in this project to trace the regulatory signal along all three-node long paths. One mutation cycle is described as follows:

1. Create a daughter network by adding or removing an interaction between two nodes, or by either doubling or losing an already existing node. Interaction (link) mutations can be attempted with average frequency f_{link}. When a link mutation is selected, it has the probability p_{ll} of being a link removal, and $1 - p_{ll}$ to be link addition.
2. If an interaction (link) is to be added, it is randomly chosen, with equal probabilities $p = 1/3$, to be an OR activation, an AND activation, or a repression.
3. Select a random Boolean input state for all nodes in the network.
4. Propagate the regulatory interactions between the nodes along all three-node-long interaction paths using simultaneous update.
5. If of both systems exhibit identical dynamics (Boolean node states), the expression pattern is preserved, and mother network is replaced with the daughter network. If the dynamics differ, we keep the mother network.

Nodes that have no regulatory interactions (in- or out-links) at the completion of a mutation cycle are removed from the network.

2.3 Network randomization with preservation of degree distribution

To randomize the evolved networks while keeping their degree distribution preserved we used the *local rewiring algorithm*, which was proposed in [Maslov, Sneppen, and Zaliznyak, 2004]. One step of this algorithm is shown in Figure 2.1. Di-

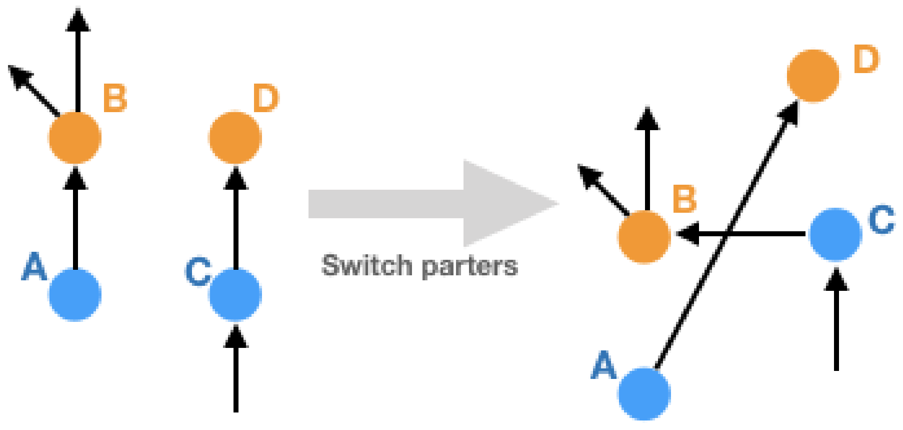

FIGURE 2.1: One step of the algorithm used to randomize directed networks with degree preservation. Two pairs of nodes, each connected to each other with a directed link, $A \to B$ and $C \to D$ are found, and the links rearranged, to obtain connected pairs: $A \to D$ and $C \to B$. This operation is allowed only if neither link $A \to D$ nor link $C \to B$ were present in the network before the switch. This algorithm conserves the in- and out-degree of each node.

rected links are swapped, pair-by-pair, until the network is randomized, while the number of links in- and out- of each node remains unchanged.

For a network with L links, after one step $L - 2$ links remain undisturbed. After n swaps, the probability that a link chosen at random is unchanged is:

$$p_{unchanged} = (1 - \frac{2}{L})^n \approx e^{-2n/L} \tag{2.1}$$

We iterated the algorithm $n \sim 1.5L$ times to generate, networks that were, on average 95% randomized but had their original degree distribution preserved.

2.4 Regulation

2.4.1 Nested Canalizing Boolean functions

The Genetic regulatory network of λ bacteriophage is compact and well studied. On Figure 1.1 (reproduced from [Trusina A, 2005]) directed regulatory interactions between the proteins in this GRN are marked with green arrows for positive (activation) and red arrows for negative regulation. Still, knowing of the logic of the interactions alone is not sufficient to determine the effective regulation if, for example CI receives activation input from CII and also repression from Cro.

In canalizing regulation, at least one of the inputs over-rides all others when it takes a specific value (known as its *canalizing value*), and thus determines the resulting regulatory action. Nested canalization is used to introduce hierarchical ordering of more than one canalizing logical input.

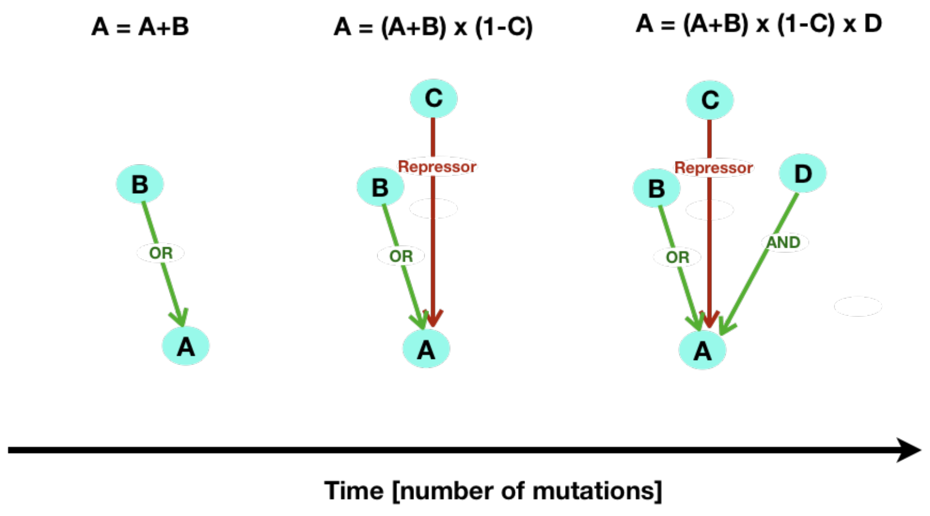

FIGURE 2.2: Nested Canalizing Boolean functions are implemented to calculate the total regulation of each node. Activation is shown with green arrows, and the red arrow represents transcription factor repression. Two types or activation are shown: one represented with the Boolean *OR*, switches the receiving node value *ON* with no preconditions, the second follows the Boolean *AND* rules. The regulatory interaction hierarchy is determined by the relative evolutionary age of the inputs.

In Figure 2.2 node *A*, receives the following regulatory inputs in order of their relative evolutionary age, from oldest to newest: *OR* activation from node *B*, repression from node C and AND activation from D. The expression state of node A is determined as follows:

$$A_{new} = (A_{old} + B) \times (1 - C) \times D \tag{2.2}$$

Constructed in this way, nested Canalizing Boolean functions are used to determine the expression state of each node while preserving the hierarchy of regulatory interactions. Simultaneous update is used as interactions are traced over all three-node long paths.

2.4.2 Regulation types

The types of regulation in the model are: AND-activation, OR-activation and repression. Regulatory input from a repressor is mathematically described as multiplication by $(1 - R)$, where R is the expression state of the repressor. Self activation of both types as well as self repression are allowed in the model.

2.5 Network matrix and regulation matrix

We use two matrices to describe the Canalizing Boolean Networks with nested AND and OR activation and repression.

The network matrix specifies the hierarchical architecture of regulatory connections between the nodes. If node A receives links from nodes B, C and D, the network matrix elements in row A and columns B, C and D are assigned non-zero values w_{ij}, corresponding to the logical order of the respective regulatory inputs. For example, if the network matrix describes input into node A with: $\{w_{ab}, w_{ac}, w_{ad}\} = \{1, 2, 3\}$, then the A's expression value is determined as: $A = B + C + D$, but if the network matrix values for node A regulatory input are: $\{w_{ab}, w_{ac}, w_{ad}\} = \{3, 2, 1\}$ then: $A = D + C + B$.

The j-th row of the network matrix describes the order of regulatory input into node j, and the i-th column lists the regulatory output of node i.

The regulation-matrix specifies the regulation type of each link in the network. The element q_{ij} of this matrix is assigned the value +1, if the link from the j-th node into the i-th node is AND-activation. If the j-th node acts as a repressor on the i-th node, the corresponding matrix value $q_{ij} = -1$, and for links that represent OR-activation $q_{ij} = 0$.

When a node has to be deleted due to a node loss mutation, or because it has lost all links, the corresponding row and column are deleted from the network matrix and from the regulation matrix. When a node duplication mutation takes place, a row and column are added to both matrices, and the duplicated node's regulatory hierarchy $w_{duplicate,j}$, $w_{i,duplicate}$ and regulation types $q_{duplicate,j}$, $q_{i,duplicate}$ are copied into their corresponding newly added row and column.

2.6 Visualization of evolved Canalizing Boolean Networks

Figure 2.3 shows a diagram of evolved Canalizing Boolean network. The regulatory interactions between the nodes are indicated with arrows, red is used for repression and green arrows represent activation. The network expression pattern is also displayed: nodes marked in orange are in the on-state, and those marked in grey are off. The node size is proportional to the total number of regulatory interactions it sends and receives.

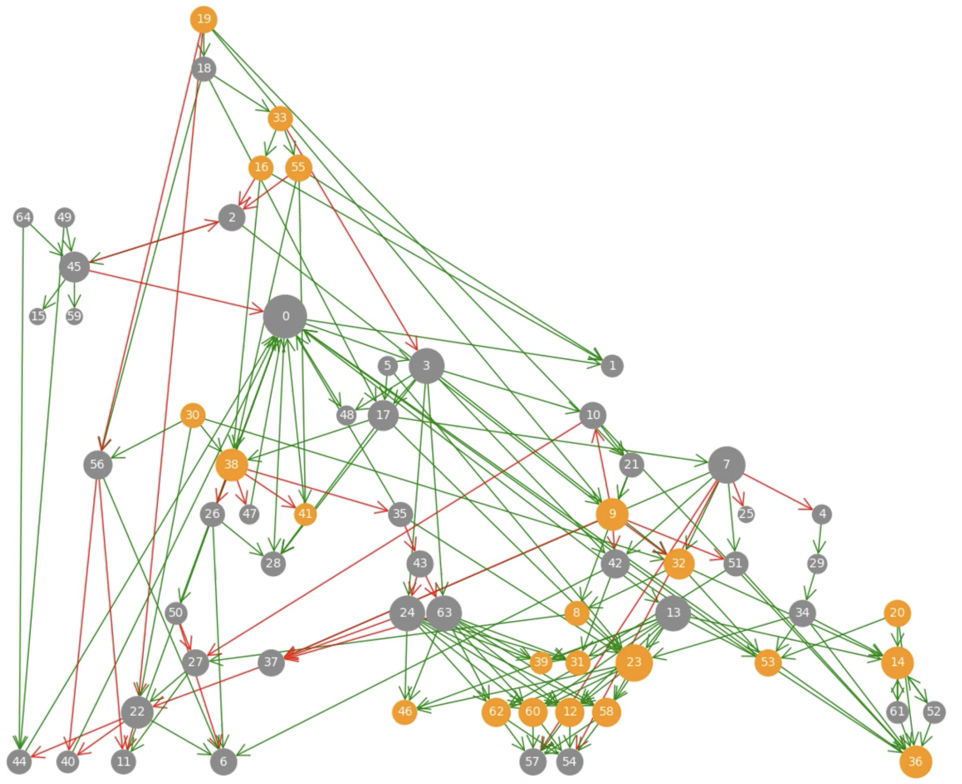

FIGURE 2.3: Diagram of evolved Canalizing Boolean network. Red arrows mark repression, and green arrows indicate activation. Nodes marked in orange are in the ON-state, and nodes marked in grey have expression state OFF

Chapter 3

Results

3.1 Network Structure

3.1.1 Link density time series

Figure 3.1 shows the time series of a Canalizing Boolean network (CBN), evolved via link-rewiring and node duplication as well as node loss mutations for which the only evolutionary selection criterion was step-wise continuity of network expression pattern, as proposed by Bornholdt and Sneppen in 1998. [Bornholdt and Sneppen, 1998] with the addition of evolutionary age-based regulatory connections prioritization.

Nested Canalizing Boolean functions were implemented to model the regulatory interactions that take place between transcription factors in Genetic Regulatory Networks (GRN). Activation regulations of two types were used, with AND and OR Boolean rules. Repression was mathematically represented with $\times(1-R)$, where R was the value of the repressor. Simultaneous update performed along all three-node-long interaction paths was used to determine the network states during each mutation cycle. Link density was measured directly from the network matrix after each mutation cycle:

$$\mathcal{L} = \frac{L}{N} \tag{3.1}$$

where N is the number of nodes in the network, and L is the number of links between them.

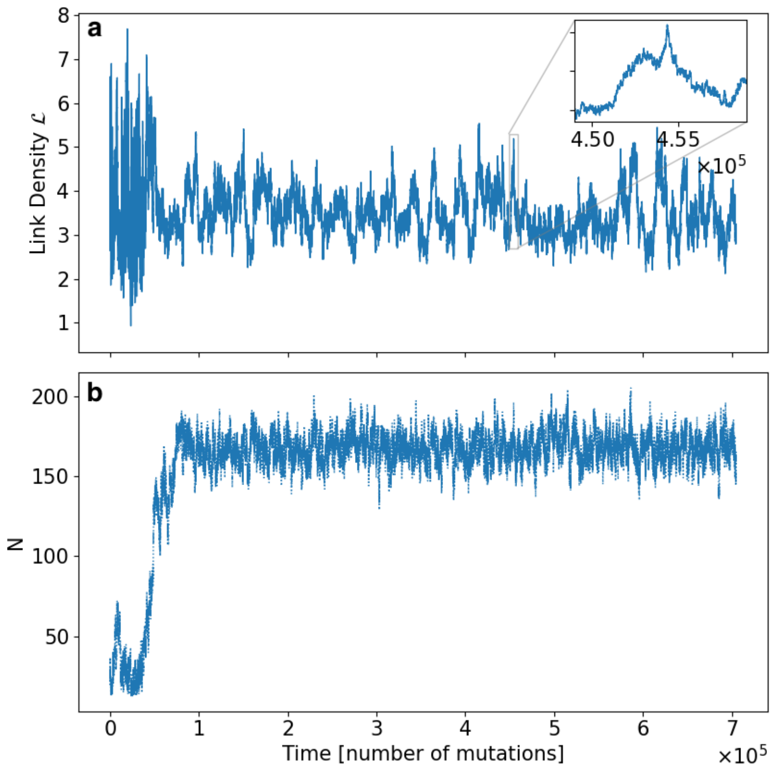

FIGURE 3.1: Long time evolution of link density of Canalizing Boolean network. (**a**) shows average link density $\mathcal{L} = L/N$ as function of time measured in the number of mutations. On (**b**), number of nodes N is plotted as a function of time. Initial network matrix was chosen randomly. Link mutations were attempted with frequency $f_{link} = 0.69$. Each link mutation was, with conditional probability of link loss $p_{ll} = 0.4$ a link removal, and with probability $p = 1 - p_{ll} = 0.6$ a link addition.
In the initial growth stage ($0 \leq Time \leq 9 \times 10^4$ mutations), network size increased stochastically from $N \sim 5$ to $N = 170$, and average link density $\mathcal{L}$ exhibited chaotic behavior. Then network growth was constrained $\langle N \rangle \sim 170$, and link density $\mathcal{L}$ was observed to enter a steady state with $\langle \mathcal{L} \rangle \approx 3.5$. N and $\mathcal{L}$ were measured directly from the network matrix.

We began with a randomly generated Canalizing Boolean network composed of $N \sim 5$ nodes. The network first underwent a rapid growth stage: link-rewiring mutations were attempted with average frequency $f_{links} = 0.69$, and the rest of the time was split equally between node duplication and node loss attempts. In Figure 3.1 this stage is shown in the first 90.000 mutation cycles. Here we observe chaotic behavior in average link density $\mathcal{L}$, and stochastic growth in the system size N.

After initial network growth, a weak constraint on the number of nodes in the system was introduced, with threshold at $\mathcal{N} = 170$. In this regime, link mutations attempts continued to occur with frequency f_{link}, but every node mutation now had a network-size dependent probability $p_{lose} \sim N/2\mathcal{N}$ to be a node loss, thus guiding the size of the network towards $\mathcal{N}$. Our motivation for limiting system size fluctuations is that the number of transcription factors in real genetic regulatory networks that we set out to model is function of complex cellular processes, and a dramatic size reduction or increase the GNR size would no doubt have disruptive consequences on the scale exceeding the scope of this project.

Nodes with all links removed at the end of a mutation cycle were deleted from the network in both stages of the CBN evolution, as they were deemed to no longer "participate" in modelled regulation interactions.

In Figure 3.1, the weak size constraint was introduced when the network first reached target size $\mathcal{N} = 170$, at approximately 90.000th mutation cycle. Network average link density in observed to then enter a "steady stage", centered at $\langle \mathcal{L} \rangle_t \approx 3.5$, where occasional dips in link density are followed by quick recovery. This 'stationary phase' is characteristic of small stochastic mutations 'local' in scope to the regulatory network – the interval of interest for this project.

3.1.2 Stationary link density time scale

For a stationary process, the autocorrelation function $\rho_{\mathcal{L}\mathcal{L}}(\tau)$ depends only on the interval length τ and can be calculated as follows:

$$\rho_{\mathcal{L}\mathcal{L}}(\tau) = \frac{\sum\limits_{t=1}^{n-\tau} (\mathcal{L}_t - \langle \mathcal{L} \rangle) \cdot (\mathcal{L}_{t+\tau} - \langle \mathcal{L} \rangle)}{\sum\limits_{t=1}^{n} (\mathcal{L}_t - \langle \mathcal{L} \rangle))} \tag{3.2}$$

Autocorrelation function $\rho_{KK}(\tau)$ of steady-state link density $\mathcal{L}$ in the evolution time series in Figure 3.1 was computed for intervals $\tau \in [1, 200.000]$, and observed to decay exponentially with τ, approximately as $e^{-\tau/T_{corr}}$, allowing us to estimate link density correlation time $T_{corr} \sim 5000$ mutations. Thus, the stationary phase link density "memory" is inferred to be of order $\sim T_{corr} \sim 5000$ mutations, and a time scale $T \sim 10^5$ is estimated sufficient to characterize this part of the link density time series. (Figure 3.2).

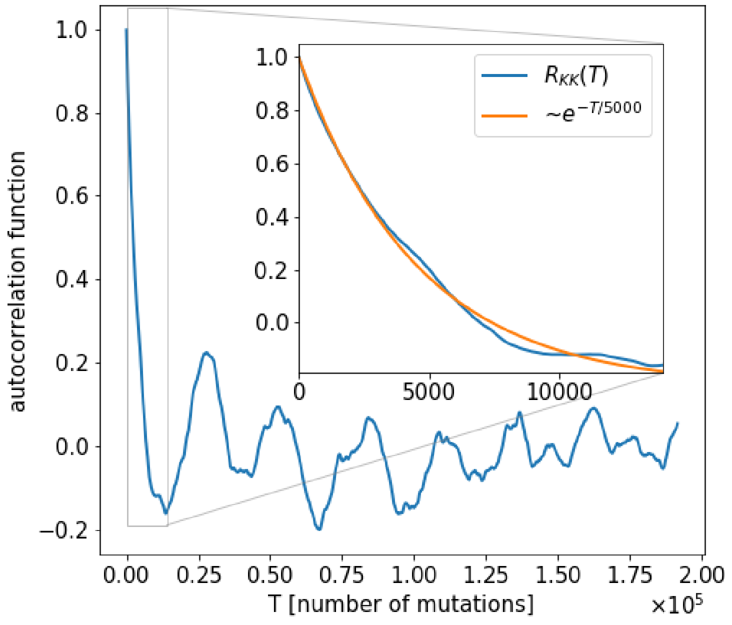

FIGURE 3.2: Autocorrelation function $\rho_{\mathcal{L}\mathcal{L}}(\tau)$ of the stationary process link density from 3.1 for lags: $1 \leq \tau \leq 200.000$ mutations. The insert shows $\rho_{\mathcal{L}\mathcal{L}}(\tau)$ in blue, and approximate fit $\sim e^{-\tau/5000}$, giving an estimated stationary process link density time scale $\sim 10^5$ mutations.

3.1.3 Distribution of average link density

Average link density $\mathcal{L} = \frac{L}{N}$, was measured directly from the network matrix after each mutation cycle. The distribution of $P(\mathcal{L})$ of link density during the stationary stage of the evolutionary process, *i.e.* the probability that the average link density of the network would equal $\mathcal{L}$, was estimated by first time-binning $\mathcal{L}$-values over the course of the "steady-state" time series. The normalized binned counts were found to be naturally approximated by a Poisson distribution

$$P(\mathcal{L}) \approx e^{-\langle\mathcal{L}\rangle} \frac{\langle\mathcal{L}\rangle^{\mathcal{L}}}{\mathcal{L}!} \tag{3.3}$$

centered in the time average of the link density $\langle\mathcal{L}\rangle$ (Figure 3.3).

The approximately Poisson form of $P(\mathcal{L}_t)$ is naturally expected for a long-time series where independent small increment changes in the number of nodes N or/and number of links, L are accepted at random time intervals.

In this section, a long-time CBN evolution via link rewiring and node duplication/loss mutations was described as a two-stage process: initial growth of the system takes

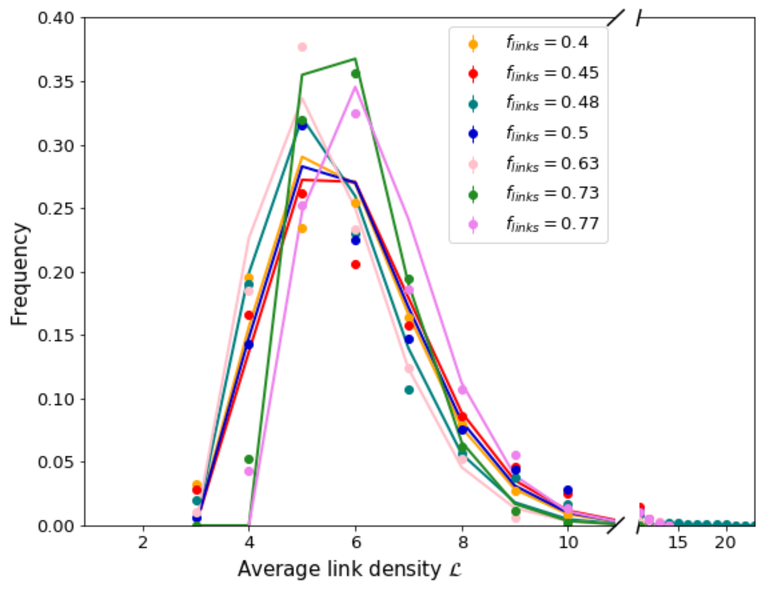

FIGURE 3.3: Distribution of average link density $P(\mathcal{L})$ for several evolutionary time series, in which link mutations were attempted with frequency $f_{links} \in [0.4, 0.77]$ is observed to fit the Poisson distribution approximately.

place in the first stage, characterized by chaotic link density, $\mathcal{L}$ and stochastic network size increase. Imposing a weak constraint on the system size fluctuations brings the network into the stationary link density stage, characterized by approximately Poisson distribution of network link density $P(\mathcal{L}_t)$. The autocorrelaton function of stationary state link density, $\rho_{\mathcal{L}\mathcal{L}}(\tau)$ was found to be naturally approximated by an exponential $e^{-\tau/T_{corr}}$, rendering an estimate for time scale of the stationary link density phase as $\sim 2 \cdot T_{corr}$.

3.2 Parameter investigation

3.2.1 System size

Quartets of Canalizing Boolean Networks with weak size constraints imposed at $\mathcal{N} = \{60, 150, 180, 250\}$ and all other parameters kept the same in each quartet, were evolved, and their steady state link density, $\langle \mathcal{L} \rangle_t$ as well as standard deviation $\sigma_{\mathcal{L}}$ were measured. Values $\langle \mathcal{L} \rangle_t \pm \sigma_{\mathcal{L}}$ were plotted with one of the series parameters, link mutation attempt frequency f_{link} along the x-axis in Figure 3.4. We observe

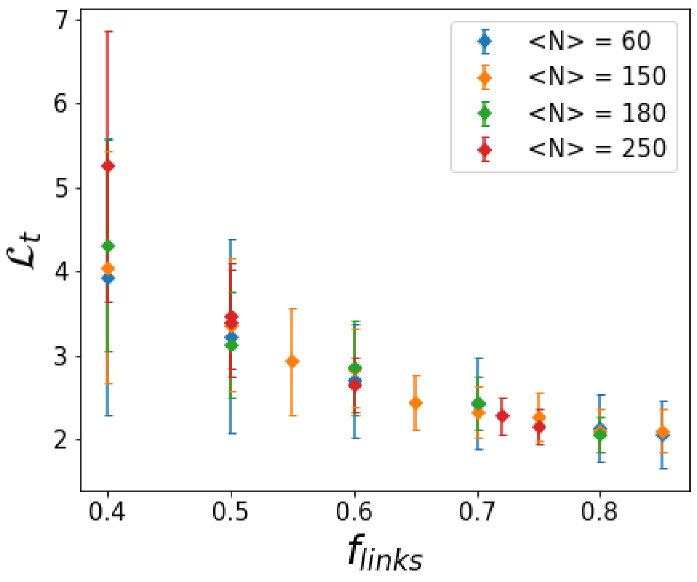

FIGURE 3.4: Time average link density $\langle \mathcal{L} \rangle_t$ measured directly from the network matrices, for CBNs with average size ranging from $\langle N \rangle = 60$ to $\langle N \rangle = 250$ is shown as function of f_{link} – the frequency with which link mutations were attempted during networks evolution. Error bars are given by standard deviation of link density, $\sigma_{\mathcal{L}}$. No dependence of link density $\mathcal{L}$ on size N of the network was observed.

no steady state link density dependence on the network size, as the corresponding average link density values with size constraints at all $\mathcal{N} = \{60, 150, 180, 250\}$ are well within standard deviations of each other.

3.2.2 Mutation acceptance probabilities

Parameters used to characterize Canalizing Boolean network in the numerical simulation stage of this project were: f_{link} – the average frequency with which link mutations were attempted during the evolutionary process, and p_{ll} – the conditional probability that a given link mutation would attempt to remove (and not add) a link. f_{link} and p_{ll} were useful during the simulation stage, but they provide no information on the network dynamics or structure – for example, one cannot deduce if any mutations got accepted at all during a CBN evolution described only by the simulation parameters f_{links} and p_{ll}. Therefore we explore a set of dynamic parameters: acceptance probability for each mutation type.

It was shown in 3.1.3 that the overall mutation acceptance, where all four types of

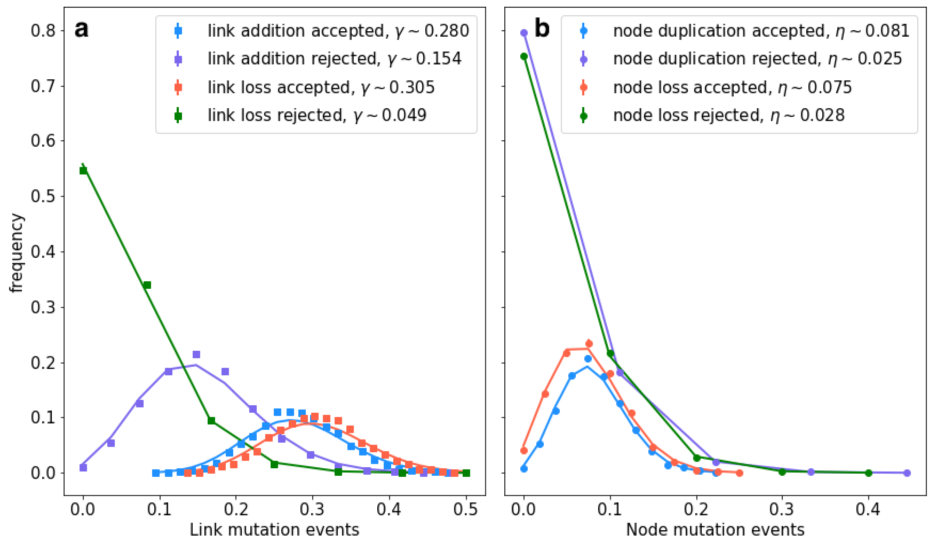

FIGURE 3.5: The types and outcomes of 250.000 mutation cycles of one Canalizing Boolean network were recorded and time-binned. (**a**) Link mutation outcome data approximately fit Poisson distributions, and corresponding probabilities were estimated from the distributions mean. (**b**) Node mutation outcome data approximately fit Poisson distributions centered at the corresponding estimated mutation probabilities. During this CBN evolution link mutations were attempted with frequency $f_{links} = 0.79$, and each had conditional probability of a link loss $p_{ll} = 0.45$.

mutation are included (link removal or addition, or node duplication or loss) is approximately a Poisson process. We expect the same to be true for all mutation types individually.

Acceptance probability for each type of mutation was estimated by first recording the type of each attempted mutation, as well as the result of the corresponding cycle (mutant network accepted or rejected). This data was obtained by time binning the activity and count of how many accepted events occurred for each bin, which naturally followed a Poisson distribution. The average of each distribution then allows us to estimate the corresponding average acceptance probability.

Figure 3.5 shows binned events counts normalized for one mutation cycle, for all types of mutations and their outcomes for one CBN evolution. The acceptance probabilities of link addition γ_+ and link removal γ_-, as well as node duplication η_+ and node loss η_- were estimated, each from the mean of the corresponding Poisson distribution. Node loss acceptance probability was observed to approximately equal that of node duplication $\eta_- \approx \eta_+$.

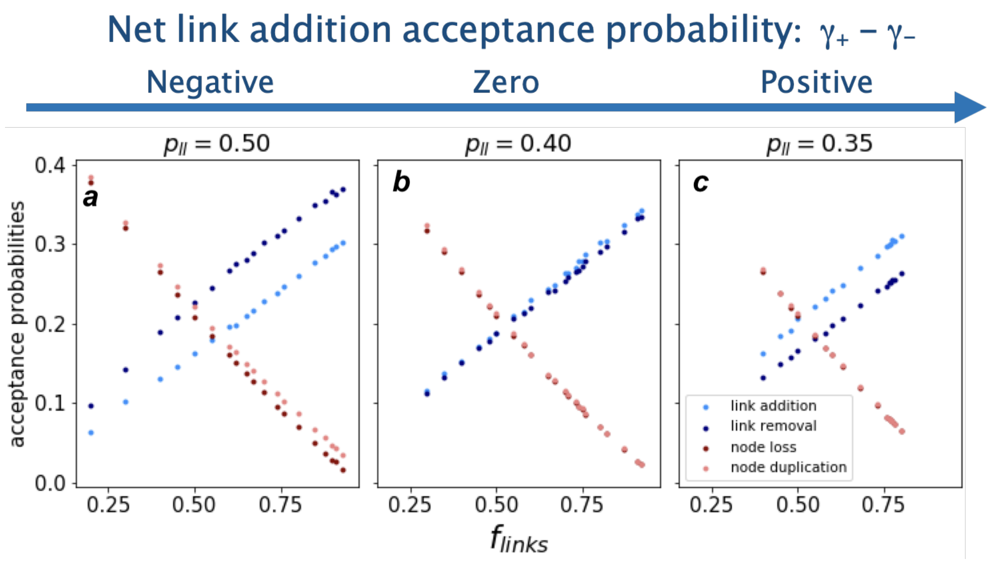

FIGURE 3.6: Mutation acceptance probabilities. Mutation acceptance probabilities estimated for networks evolved with conditional probability of link loss $p_{ll} = 0.5$ are shown on (**a**), with $p_{ll} = 0.4$ are shown (**b**) and for those evolved with conditional probability of link loss $p_{ll} = 0.35$ on (**c**). Frequency of link mutation attempts during the CBN evolution f_{links} is shown along the x-axis on all three panels. Net link addition acceptance probability $\gamma_+ - \gamma_-$ is negative on (**a**), equals zero on (**b**) and is positive on (**c**).

Figure 3.6 shows the mutation acceptance probabilities estimated for 95 CBNs as function of link mutation attempts frequency, $f_{link} \in [0.2, 0.96]$. For all parameters, node loss acceptance probability was observed to approximately equal node duplication acceptance probability, $\eta_+ \approx \eta_-$, and both will occasionally be referred to as η. Networks exhibited positive net link addition acceptance probability $\gamma_+ - \gamma_- > 0$ if they evolved with low conditional link loss probability, $p_{ll} = 0.35$, as shown on (a), had zero net link addition acceptance $\gamma_+ - \gamma_- = 0$, when any attempted link mutation had probability $p_{ll} = 0.4$ of being a link removal in (b) and had negative net link addition acceptance, when half of link rewiring attempts during its evolution were link addition, in panel (c):

p_{ll}	$\gamma_+ - \gamma_-$
0.35	positive
0.4	zero
0.5	negative

We show later that structures (in 3.2.4) and expression pattern (in 3.4) of Canalizing

Boolean networks evolved with positive or zero net link addition differ from those of networks for which the net probability of link addition acceptance was negative during the evolutionary process.

3.2.3 Link to node mutation acceptance ratio

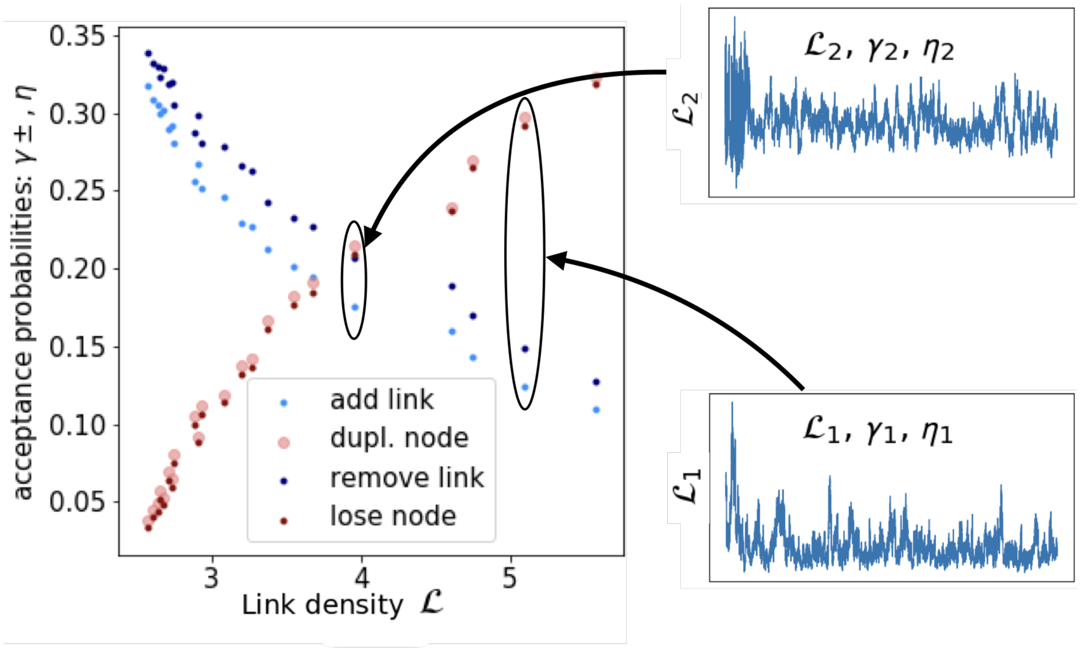

FIGURE 3.7: Acceptance probabilities for link rewiring $\gamma_\pm$ and for node duplication η mutations estimated from evolution time series of 25 CBNs. Each γ_+, γ_-, η triplet is shown with the average link density $\mathcal{L}$ of the corresponding network along the x-axis. We observe node duplication acceptance probability η is highest for networks with largest average link density $\mathcal{L}$. Networks evolved with higher probability of link rewiring acceptance $\gamma_\pm$, have lower average link density $\mathcal{L}$.

Mutation acceptance probabilities $\gamma_\pm$ and η estimated in 3.2.2, are plotted on Figure 3.7 with the corresponding link density $\mathcal{L}$ along the x-axis. Observe that networks evolved with high probability to accept node duplications and low probability of link rewiring acceptance reach highest link densities $\mathcal{L}$. Networks that had lower $\mathcal{L}$ were evolved accepting link mutations more often and had lower probability of acceptance for node duplication.

This can be understood as follows: when a link addition is accepted, the number of links is incremented by one: $L \to L+1$, and the link density of the network is increased by: $\Delta\mathcal{L} = 1/N$. When a node is duplicated, the number of nodes in the network is incremented by one: $N \rightarrow N+1$, and the number of links: $L \rightarrow L + \Delta l$ where Δl is the number of links into plus the number of links out of the duplicated node, increasing the link density by: $\Delta\mathcal{L} \approx \Delta l / N$ where and $\Delta l \geq 1$. Furthermore, at first approximation, each node duplication attempt is expected to add, on average, $\mathcal{L}$ links to the network: $\langle \Delta l \rangle \sim \mathcal{L}$, as the node to be duplicated is selected at random. However, we are not able determine at this point if more connected nodes are accepted at the same, higher or lower rate than less connected ones.

To investigate how the relationship between link rewiring and node duplication acceptance probabilities affects the evolved network structure, we introduce link-to-node mutation acceptance ratio v, which quantifies how many link mutations are accepted, on average, for each accepted node duplication:

$$v = \frac{\gamma_+ + \gamma_-}{\eta} \tag{3.4}$$

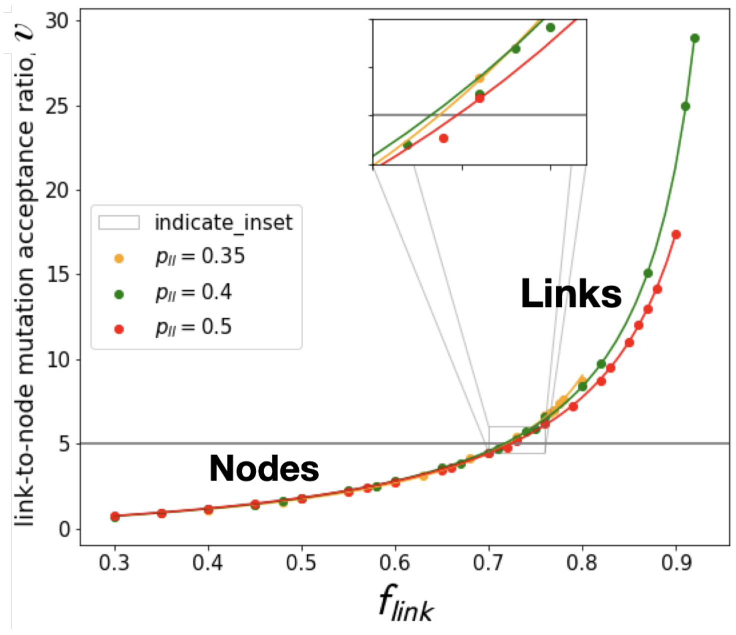

FIGURE 3.8: Link-to-node mutation acceptance ratio $v = \frac{\gamma_+ + \gamma_-}{\eta}$ was estimated from evolution time series of 45 CBNs and plotted along the y-axis, with frequency of link rewiring attempts f_{link} of corresponding networks on the x-axis. Large values of v correspond evolutionary dynamics driven by link mutations, whereas small v indicate node duplication/loss was dominant during the network evolution.

In Figure 3.8 link-to-node mutation acceptance ratio v was estimated for 45 different Canalizing Boolean network evolution time series and plotted against the average frequency of link mutation attempts f_{link} for corresponding networks. We observe that for large link-to-node mutation acceptance ratio v, small perturbations δf_{link} lead to large changes Δv. For small $v \leq 2$, the opposite is true: small changes to frequency of link mutation attempts δf_{link} have negligible effect on v. In other words link-to-node mutation acceptance ratio v was small for networks whose evolution was dominated by node mutations, whereas for networks formed primarily via link mutations $v \geq 5$ was true.

3.2.4 Approximate average link density model

Figure 3.9 shows average link density as function of link-to-node mutation acceptance ratio $\mathcal{L}_t(v)$ introduced above in 3.2.3. 45 Canalizing Boolean networks were evolved with link mutation frequency $f_{link} \in [0.2, 0.96]$ and with one of three different conditional probabilities of link loss, $p_{ll} = \{0.35, 0.4, 0, 5\}$. Link density $\mathcal{L} = L/N$, where L is the number of links, and N is the number of nodes in the network, was measured directly from the network matrices during the evolution. Standard deviation $\sigma_{\mathcal{L}}$ was calculated for each $\mathcal{L}_t$ series. Mutation acceptance probabilities for links ($\gamma\pm$), and for nodes (η) were estimated from each networks evolutionary time series as described in 3.2.2, and link-to-node mutation acceptance ratio $v = (\gamma_+ + \gamma_-)/\eta$ was obtained for each Canalizing Boolean network evolution. Now the evolution time series of each network can be characterized in terms of how many link rewiring mutations are accepted, on average, for each accepted node duplication.

For all networks, for very small v, where 30% or more of accepted mutations were node duplication, steady state link density started high and appeared to approximately follow $\mathcal{L} \sim 1/v$. In networks evolved with non-negative net link addition acceptance probability, $\gamma_+ \geq \gamma_-$, contributions by link mutations overtook those of by node duplication as v increased, and the decline in link density was reversed by $\mathcal{L} \sim v$ dependency at $\mathcal{L}_{min} \sim \gamma_+/\gamma_-$. This behavior is described by the approximate

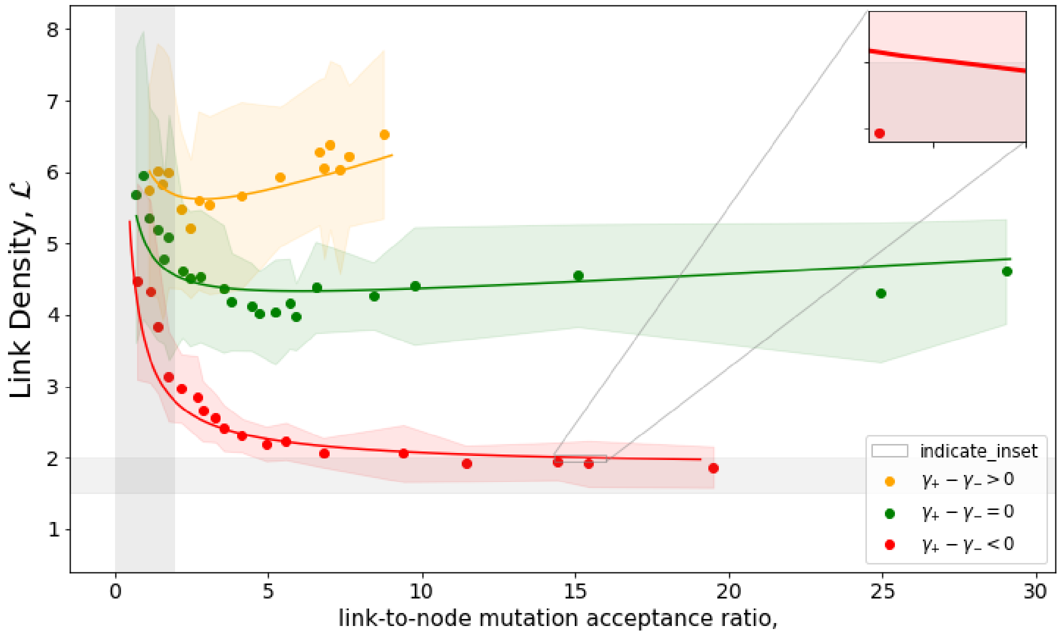

FIGURE 3.9: Average link density $\mathcal{L}_t$ as a function of link-to-node mutation acceptance ratio $v = \frac{\gamma_+ + \gamma_-}{\eta}$. Standard deviation $\sigma_{\mathcal{L}}$ was calculated directly from the evolutionary time series for each $\mathcal{L}_t$, and marked with either yellow, red or green shading.
In the vertical region shaded in grey, node mutations are dominant, in the intermediate region, $2 \leq v \leq 5$, link mutations contribute as well. For $v \geq 5$, link mutations dominate.
Points in yellow represent $\mathcal{L}_t$ of networks evolved with $p_{ll} = 0.35$, points in green had $p_{ll} = 0.4$, and those in red evolved with $p_{ll} = 0.5$. Only red data points correspond to CBNs with estimated negative net link addition probability $\gamma_+ - \gamma_- < 0$.
The curves in yellow and in green are obtained with the $\mathcal{L}_t(v, \rho)$ approximate model in equation 3.5, the red curve is from equation 3.6. The horizontal region shaded in gray shows the approximate critical connectivity zone for the CBNs

model:

$$\mathcal{L}(v,\rho) \approx v \cdot \left[1 - \frac{1}{\rho}\right] + \frac{1}{v} + 4\rho \quad where \quad \rho = \frac{\gamma_+}{\gamma_-} \tag{3.5}$$

For networks generated with $\gamma_+ - \gamma_- < 0$, link density did not experience recovery, and continued to decline as v grew:

$$\mathcal{L}(v,\rho) \approx \frac{1}{v} \cdot \left[\frac{1}{1-\rho^2}\right] + \rho + 1 \quad where \quad \rho = \frac{\gamma_+}{\gamma_-} \tag{3.6}$$

On figure 3.9 the curve in yellow, with $p_{ll} = 0.35$, corresponds to networks evolved with positive net link addition acceptance probability $\gamma_+ - \gamma_- > 0$, the green curve, with $p_{ll} = 0.4$, represents systems evolved with $\gamma_+ - \gamma_- = 0$. These curves were obtained with model in equation 3.5. In the region $v > 4$ the term linear in v dominates in equation 3.5, reducing it to:

$$\mathcal{L} \approx v \cdot const \tag{3.7}$$

The solid curve in red, for $p_{ll} = 0.5$, follows the model for networks with negative net link addition acceptance probability in 3.6.

We examined structural characteristics and external parameters of evolutionary timelines of 45 Canalizing Boolean network evolutions (average link density $\mathcal{L}_t$, number of nodes N, frequency of link mutations, f_{link}, conditional probability of attempting a link removal, p_{ll}). Dynamic parameters were estimated to describe internal processes in these networks.

First, we estimated mutation acceptance probabilities for link rewiring $\gamma_\pm$, and for node duplication/loss $\eta_\pm$ by finding the mean of approximately Poisson distributions of counts of mutation events and their outcomes. Then we observed that average link density $\mathcal{L}$ was highest for networks with largest node mutations acceptance probability, and introduced link-to-node mutation acceptance ratio $v = (\gamma_+ + \gamma_-)/\eta$, to quantify how many link rewirings were accepted, on average, for each accepted node duplication during the network's evolution. Finally, a approximate toy model for describing the networks average link density in terms of dynamic parameters, $\mathcal{L}(v, \frac{\gamma_+}{\gamma_-})$ was suggested.

3.3 Degree Distribution

3.3.1 Average degree distribution

This section is focused on the topological features of evolved Canalized Boolean networks. First, we estimated the average degree distribution during long-time evolution of a network for which both number of nodes N and number of links L vary. The in-, out- and total-degree for each node was calculated directly from the network matrix after each mutation cycle. Degree counts were aggregated for each network size N, and count of network size recurrence was kept. Time series degree count totals for each N were divided by corresponding network size recurrence count. To enable comparison of these data for different network sizes, degree counts were divided by the corresponding number of nodes N. The resulting distributions can be seen in

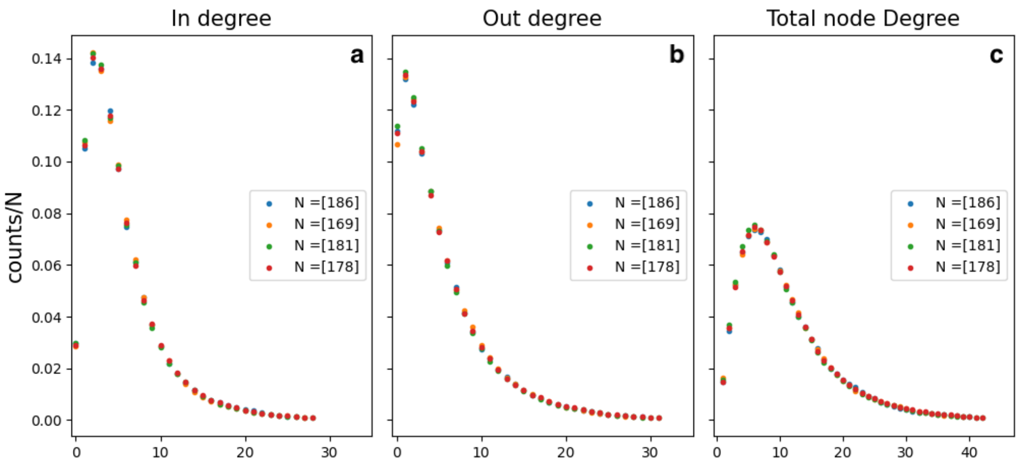

FIGURE 3.10: The in-degree (**a**), out-degree (**b**) and total degree = in-degree + out-degree (**c**) counts for CBN with link-to-node mutation acceptance ratio $v = 4.3$ is shown for four of the 20 most recurring network sizes N. Degree counts were aggregated for each network size N over the evolutionary time series, then divided by the recurrence count of N, and normalized by the corresponding N. No significant difference is observed between In-degree and Out-degree counts. No significant difference is seen between degree distributions at $mode(N_t) = 178$ and one at the 20th most frequent network size, $N = 186$.

figure 3.10 for networks which had, during their evolution a link-to-node mutation acceptance ratio $v = 4.3$ and network size constrained to fluctuate around $N = 180$ nodes. No significant discrepancies between the in-degree and the out-degree are seen, characterizing the evolved Canalizing Boolean networks structure as, on average, in-out degree symmetric or nearly symmetric. Degree frequencies at $mode(N_t)$ = 179 were observed to not deviate significantly from corresponding frequencies at the fifth, 10-th and 15-th and 20-th most attained network sizes, therefore we continue to use only the counts at $mode(N_t)$ to estimate average degree distribution of CBN long-time evolution.

All three degree distributions (in-degree, out-degree, and total = in-degree + out-degree) on figure 3.10 exhibit fat tails.

Let's discuss the degree distributions expected for random boolean networks. Erdos-Reynei (ER) model [Erdös and Rény, 1960] is the simplest description of a random network with N nodes constructed by connecting each node pair with probability p. The (ER) network has average connectivity:

$$K = p(N-1) \tag{3.8}$$

and, for large N, an approximately Poisson degree distribution:

$$P(k) \approx e^{-\langle k \rangle} \frac{\langle k \rangle^k}{k!} \tag{3.9}$$

We expect to see Poisson degree distribution in randomly constructed Boolean networks. However, we observe degree distribution with fat tails, and therefore wider than Poisson for the Canalizing Boolean networks evolved here with step-wise preservation of the expression pattern and logical signal tracing along all three-node long paths.

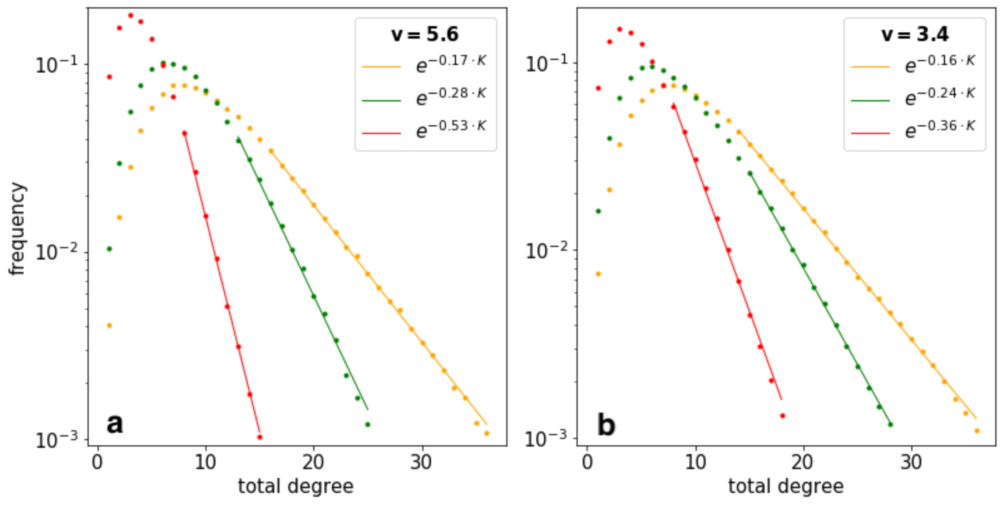

FIGURE 3.11: Degree distribution (in-degree + out-degree) $P(k)$ of networks evolved with $v = 5.6$ in (**a**), and with $v = 3.4$ in (**b**), and probability $p_{II} = 0.35$ in yellow, $p_{II} = 0.4$ in green, and $p_{II} = 0.5$ – in red. Resulting distributions $P(k)$ are approximated with an exponential, for $k \geq 8$.

To investigate the evolved networks degree distribution further, we considered six evolved Canalizing Boolean networks: three were evolved with approximately 5.6 link rewirings accepted for each node duplication acceptance ($v = 5.6$), and their estimated degree distributions are shown on (a) in Figure 3.11. The three networks whose average degree distributions are plotted on (b) were evolved with $v = 3.4$. On both panels, the curve in red evolved with negative net link addition acceptance probability, in green with $\gamma_+ = \gamma_-$, and in yellow with positive $\gamma_+ - \gamma_-$. Only total degree distribution is shown as in-out-degree symmetry or near symmetry was observed for degree distribution of these Figure 3.10. All six curves exhibited exponential tails for $k \geq 8$:

$$P(k) \sim \tau \cdot e^{-\tau k} \;\; with: \;\; 0.16 \leq \tau \leq 0.53 \tag{3.10}$$

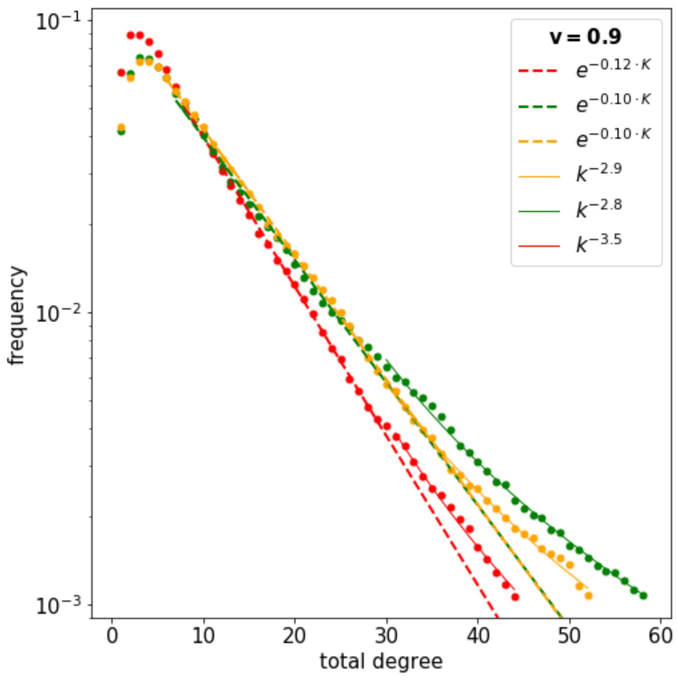

FIGURE 3.12: Degree distribution (in-degree + out-degree) $P(k)$ for networks evolved with link-to-node mutation acceptance ratio $v = 0.9$ and $p_{ll} = 0.35$ in yellow, $p_{ll} = 0.4$ in green, and $p_{ll} = 0.5$ – in red. Resulting distributions $P(k)$ are approximated with exponential curves with means at corresponding $2\mathcal{L}$, and exhibit short power tail $k \sim k^{-\alpha}$, $\alpha \approx 3$ for degrees $k > 35$.

The exponential tails had the mean approximately equal to double the corresponding networks link density:

$$\langle \tau \cdot e^{-\tau \cdot k} \rangle = \frac{1}{\tau} \approx 2 \cdot \mathcal{L}(v) \tag{3.11}$$

This is expected, as the degree of a node is determined by counting the number of links coming in and out of it. One link connects two nodes and the average degree:

$$\frac{\sum_0^N (k_{degree})}{N} = \frac{2L}{N} = 2 \cdot \mathcal{L} \tag{3.12}$$

Furthermore, on figure 3.12 we show that networks which accept on average one link rewiring mutation for each accepted node duplication ($v \sim 1$) had approximately exponential average degree distribution, $P(k) \sim \tau \cdot e^{-\tau K}$ with the mean $\tau \sim 1/2\mathcal{L}$, where $\mathcal{L}$ is the corresponding average link density, and with short power tails $P(k \geq 35) \sim k^{-\alpha}$, with $\alpha \sim 3$.

Exponential degree distribution was observed in real genetic networks.

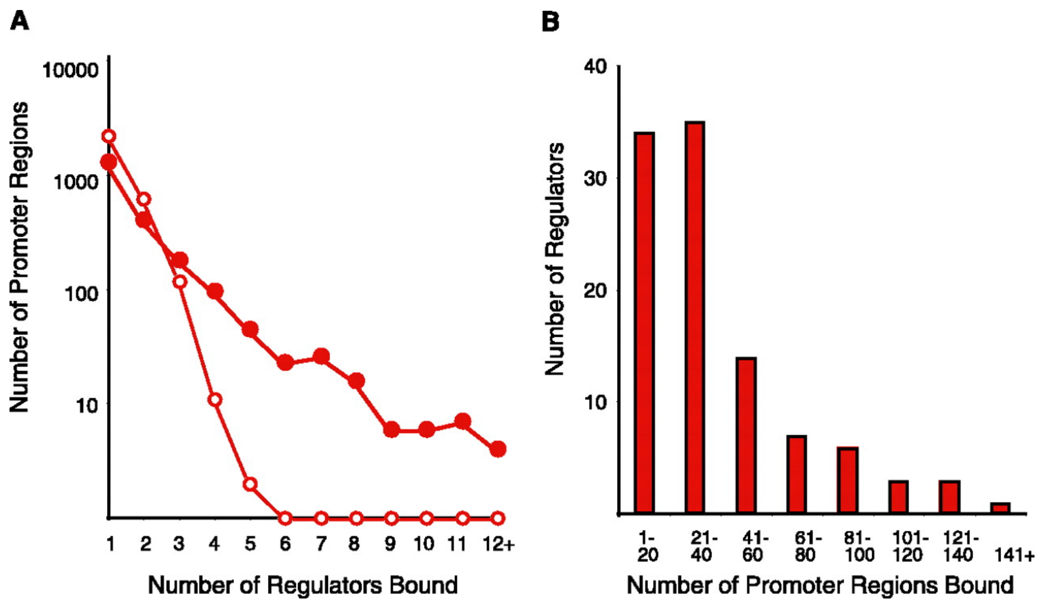

FIGURE 3.13:]
Genome-wide distribution of transcriptional regulators (reproduced from [Lee et al., 2002]). (**A**) Plot of the number of regulators bound per promoter region. The distribution for the actual location data (red circles) is shown alongside the distribution expected from the same set of P values randomly assigned among regulators and intergenic regions (white circles). At a P value threshold of 0.001, significantly more intergenic regions bind four or more regulators than expected by chance. (**B**) Distribution of the number of promoter regions bound per regulator..

Figure 3.13 was reproduced from [Lee et al., 2002] and shows genome-wide degree distributions of transcription regulation in *Saccharomyces cerevisiae*. On (A) the number of regulators bound per promoter region is shown to be exponentially distributed with a possible power tail. In our model "regulators bound per promoter region" translates as in-degree, and we find exponential distribution (or exponential-tailed, depending on the value of v) for both in-degree and out-degree.

(B) Shows the distribution of number of regions bound per regulator, which in our evolution model is represented by out-degree. The distribution of existing regulatory interactions in *Saccharomyces cerevisiae* in panel (B) resembles a Poisson distribution, not an exponential or exponential-tailed as found for out-degree in our model output.

A study of signaling in signaling in *Escherichia coli* protein reaction network [Axelsen, Krishna, and Sneppen, 2007] found the in-degree distribution of 1938 reaction nodes (including metabolic, complex-forming, and 812 transcription regulation) to be exponential while the out-degree for same genetic network followed a power-distribution .

That our results do not fully replicate the findings of the genome-wide location analysis in of *Saccharomyces cerevisiae* [Lee et al., 2002] and of the 1938-node weak giant component of *Escherichia coli* genetic network [Axelsen, Krishna, and Sneppen, 2007] is not unexpected, as our model is focused on regulatory interactions between transcription factors only, a scope much narrower than genome-wide.

The evolved Canalizing Boolean networks did not exhibit degree distribution expected for random Boolean networks (Poisson), and instead were observed to have a wider degree distribution with exponential tails. Furthermore, networks accepting on average one link rewiring mutation for each node duplication had exponential degree distributions with power tails for $k > 35$ with exponent $\alpha \sim 3$.

Real Genetic Regulatory networks with exponentially distributed in-degree reaction-nodes had been observed [Lee et al., 2002], [Axelsen, Krishna, and Sneppen, 2007]. Same studies also found the corresponding out-degree to be power-distributed, but the wider observed out-degree could be due to the wider scope of the studies, which included metabolic, complex-forming and transcription regulation, whereas the Canalizing Boolean networks in our project are models for transcription factor-only regulation networks.

3.3.2 Dispersion of average degree distribution

We have discussed the average long-time degree distribution of evolved CBNs, and found it to be wider than expected for random Boolean networks. Now it would be of interest to consider a finer structure of this distribution. Fano factor (FF) and coefficient of variation (CV) are used to quantify dispersion in a distribution with variance σ^2 and mean μ:

$$FF = \frac{\sigma^2}{\mu} \tag{3.13}$$

$$CV = \frac{\sigma}{\mu} \tag{3.14}$$

From the definition, for exponential distribution, $p(k) = \mu \cdot e^{-\mu \cdot k}$, for which $\sigma = \mu^{-1}$:

$$CV_{exponential} = 1 \quad and \quad FF_{exponential} = \frac{1}{\mu} \tag{3.15}$$

Similarly, for Poisson distribution, with $\sigma_k^2 = \mu$:

$$CV_{Poisson} = \frac{1}{\sqrt{\mu}} \quad and \quad FF_{Poisson} = 1 \tag{3.16}$$

Fano factor and coefficient of variation were computed directly from the degree counts (used previously to estimate the degree distribution) for approximately 50 evolved CBNs, and plotted as function of link-to-node mutation acceptance ratio $v = (\sigma_+ + \sigma_-)/\eta$ on (a) and (c) respectively in Figure 3.14. FF and CV are plotted as function of average link density $\mathcal{L}$ on (b) and (d) respectively.

Fano factor is observed to asymptotically approach $FF_{Poisson} = 1$ for networks with $v \to \infty$. For smaller v, Fano factor of the evolved Canalizing Boolean networks degree distribution grew rapidly. In networks evolved accepting on average one link mutation for each node duplication ($v \sim 1$), Fano factor of degree distribution

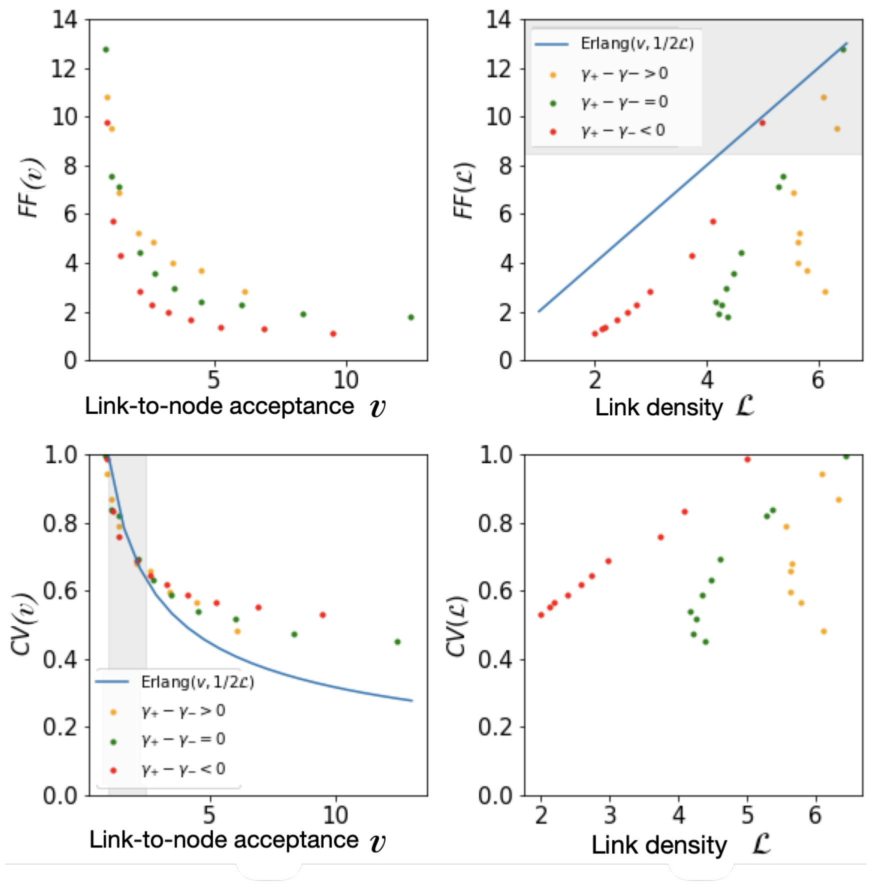

FIGURE 3.14: Fano Factor (FF) and coefficient of variation (CV) computed directly from degree counts, and plotted on (**a**) and (**c**), as function of link-to-node mutation acceptance ratio v, which describes the average number of link mutations accepted for each node duplication. (**b**) and (**d**) show FF and CV respectively as function of average link density $\mathcal{L}$. FF and CV for Erlang distribution with scale parameter $\mu = 1/2\mathcal{L}$ and shape parameter = v is plotted in on **c** and **c**. On (**b**) and (**c**), the region where node mutation acceptance dominates is shaded in gray.

reaches $FF_{exponential} \sim 2 \cdot \mathcal{L}$, for where $\mathcal{L}$ is average link density of corresponding time series.

For large v the coefficient of variation reaches just below $CV = 0.5$. As link-to-node mutation acceptance ratio lowers CV becomes larger reaching $CV_{exponential} = 1$ for degree distributions of CBNs evolved with link mutation and duplication having equal average acceptance probabilities (with $v \approx 1$).

Degree distributions of networks evolved with link-to-node mutation acceptance ratio $v \sim 1$ have Fano factor and coefficient of variation approaching those of exponential distribution with mean $\mu = 1/2\mathcal{L}$. However, FF asymptotically approaches the Fano Factor of a Poisson distribution for $v \to \infty$ and $CV \sim 0.5$ at large v

The behavior described above is in part replicated by the Fano factor and coefficient of variation of the Erlang distribution ["Traffic and Queueing Theory" 2008]. The Erlang-k distribution is composed of k exponential distributions that are identical and independent of each other, and is given by:

$$f(t) = \mu \frac{(\mu t)^{k-1}}{(k-1)!} e^{-\mu t} \tag{3.17}$$

k is called the shape parameter, and μ is the scale parameter. Erlang mean is equal to k/μ, and Erlang variance is k/μ^2. The Fano factor and coefficient of variation for Erlang distribution with shape parameter equal to the link-to-node mutation acceptance ratio v, and scale parameter $\mu = 1/2\mathcal{L}$, where $\mathcal{L}$ is the average link density of the corresponding network is plotted in panels (b) and (c), respectively, and approximately overlaps the FF and CV calculated for degree distributions of CBNs with $v \leq 3$.

For smaller v, we suggest that the node degree distribution for the evolved CBNs could be described as the Erlang-k distribution, with the shape parameter equal to the number of link rewiring mutations per node duplication accepted v. The larger the $k_{shape} = v$, the more frequent "customizing" link-rewiring adjustments are applied to the duplicated nodes, the narrower and less "bursty" the node degree distribution becomes. As v increases, degree distribution of the CBN transitions from essentially exponential, as for $v \sim 1$ in figure 3.12 to exponential-tailed, as in figure 3.11 for $v > 3$. When link rewiring begin to dominate the evolution process the dispersion of node degree distribution for different net link addition acceptance probability $\gamma_+ - \gamma_-$ diverges, as seen in (a) and (c) on figure 3.14.

The dispersion of estimated degree distributions for 50 evolved Cananlizing Boolean networks was studied by evaluating the Fano factor and coefficient of variation of the degree distributions. The FF and CV calculated for degree distribution of networks that accepted between one and three link rewiring for each node duplication were found to be similar to FF and CV of Erlang-k distribution with the shape parameter defined by how many link rewiring mutations are accepted per node duplication ($k_{shape} = v$), and the scale parameter inversely proportional to networks average link density. When link rewiring dominated the evolution, the dispersion of node degree distribution for different net link addition acceptance probabilities $\gamma_+ - \gamma_-$ diverged.

The topological characteristics of the evolved Canalizing Boolean networks (degree distribution and dispersion of the degree distribution) did not match with the expectation for random Boolean networks (Poisson distribution, and $FF_{Poisson} = 1$, $CV_{Poisson} = \mu^{-1/2} \approx \sqrt{2K}$), but described a distribution that is exponential for networks evolved with small link-to-node mutation ratio, and exponential-tailed for networks that accept three or more links rewiring mutations for each accepted node duplication.

3.4 Expression pattern of the evolved networks

We discussed the structural properties of Canalizing Boolean networks evolved with preserving continuity of network expression pattern. Let us now shift focus to the expression pattern.

Figure 3.15 shows expression of transcription factors of the gene regulatory network of *Saccharomyces cerevisiae* in response to various stimuli, and was reproduced from the study of *S. cerevisiae* regulatory network topology done using the Gene Ontology Consortium annotations [Axelsen, Bernhardsson, and Sneppen, 2008]. We observe:

1. for all of the stimuli pictured, there was a large component that remained unexpressed (black component in the center and top left)
2. multiple loci of the network were expressed for the majority of the stimuli.

Whether network hubs with persisting expression pattern as those on 3.15 are essential for stable gene regulation, or are natural consequence of evolutionary complexity is an interesting question.

Let's examine the "expression pattern" of Canalizing Boolean networks evolved in this project. The part of a Boolean network composed of all the nodes that do not switch in response to a change in the external (or internal) conditions is known as Frozen component [Kauffman, 1990]. In the framework of our model, the external conditions are represented by the random values assigned to all nodes at the start of each mutation cycle – the input state. Changes to internal conditions occur each time the network structure is replaced by a newly accepted mutant.

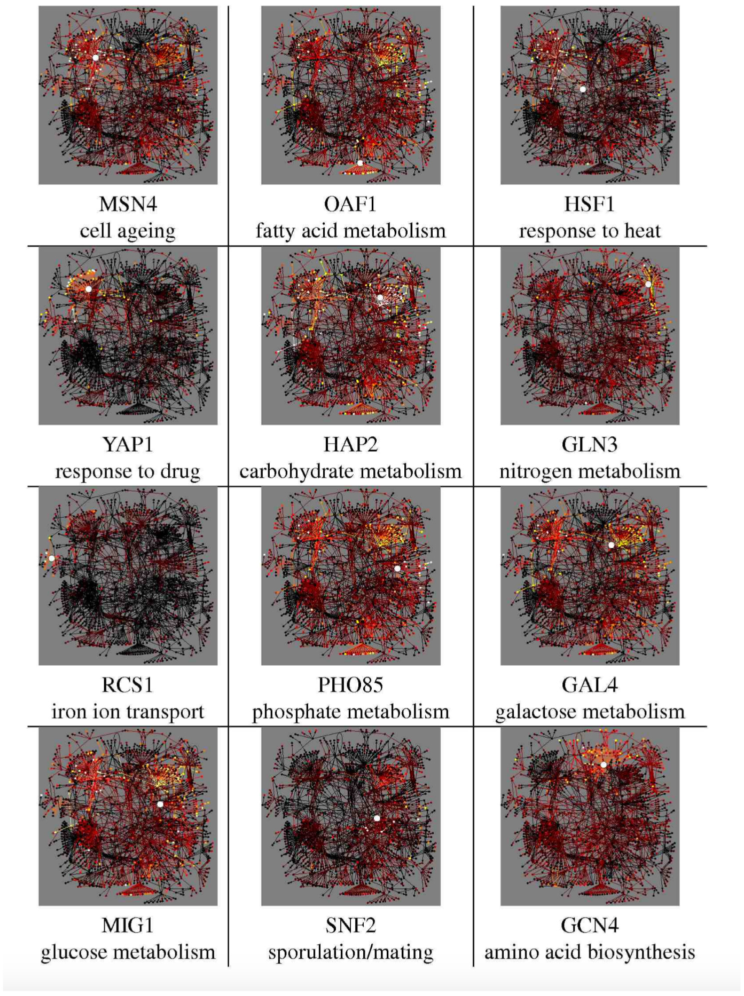

FIGURE 3.15: Responses in protein expression of the gene regulatory network of *Saccharomyces cerevisiae* to various stimuli. The response appears to be locaized, with some regions activated in response to almost all stimuli, and some locations remain un-expressed throughout (the black hubs in the center and on the left). The figure is reproduced from [Axelsen, Bernhardsson, and Sneppen, 2008]

We measured the frozen component directly from the network matrix by counting all nodes that remain in the same state during a mutation cycle repeated for thirty randomly generated input states. Testing with thirty random input states was determined sufficient to verify the stability of node expression state, and increasing this number did not substantially reduce frozen component size fluctuations. Frozen node count was normalized by the corresponding network size after each mutation cycle.

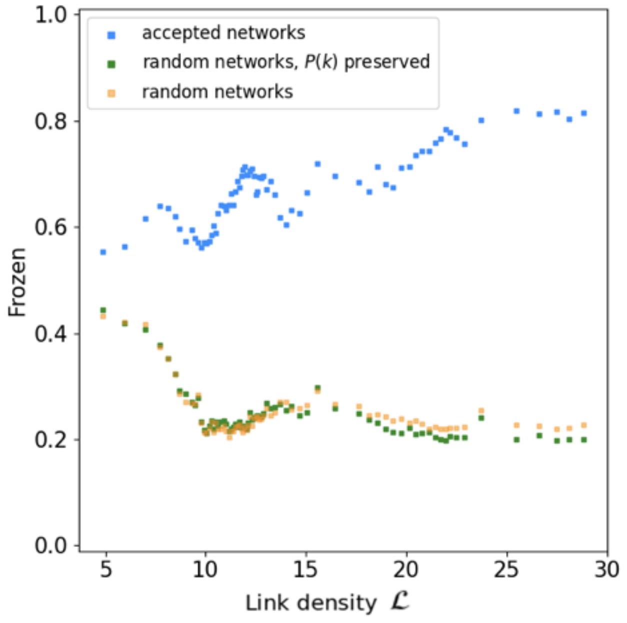

FIGURE 3.16: Frozen component of CBN evolved for 40.000 steps with link-to-node mutation acceptance ratio $v \approx 4$ and positive net link acceptance, compared to frozen component of random networks. Frozen component was measured for all newly accepted mutations of CBN, and for 2 random networks: a random network with the same N, L and degree distribution $P(k)$ as the accepted CBN mutation, and a random network with same L and N, but with links distributed randomly among the nodes. The measured frozen components were binned with respect to network link density $\mathcal{L} = L/N$, and the average in each bin plotted as function of $\mathcal{L}$ corresponding to the center of the bin: in blue - frozen component of the accepted network mutants, in green – frozen component of corresponding random networks with the same L, N, and $P(k)$ as the CBN mutant, and in yellow: of random network with same L and N only.

Frozen component was measured for for 40.000 evolution steps for all newly accepted mutations of a Canalizing Boolean network. For each accepted CBN mutant, two random networks were generated: one with the same number of links L and number of nodes N as the accepted mutant, but with the links distributed randomly among the nodes, and one random network with the same L, N, and same degree distribution as the evolved CBN. The latter was obtained from the evolved CBN via a *local rewiring algorithm* proposed by [Maslov, Sneppen, and Zaliznyak, 2004]. Frozen components were measured for both random networks as well.

Figure 3.16 shows the average frozen component as function of average link density $\mathcal{L}$ for a Canalized Boolean network evolved with link-to-node mutation acceptance ratio $v \approx 4$ and positive net link mutation acceptance $\gamma_+ - \gamma_- > 0$ (in blue). The frozen component of a random Boolean network with the same degree distribution as the evolved CBN is shown in yellow. In green is the frozen component for a random Boolean network, with random degree distribution.

We observe that the average evolved CBN frozen component is larger than the average frozen component for corresponding random networks. Between 55% and 80% of all nodes of the evolved CBN in 3.16 are observed to be continuously activated or continuously off while a mutation cycle is repeated for thirty random states. Furthermore, the frozen component the evolved CBN is higher when link density $\mathcal{L}$ is higher, but the frozen components of random networks appears to stagnate at $\sim 20\%$ for $10 \leq \mathcal{L} \leq 30$.

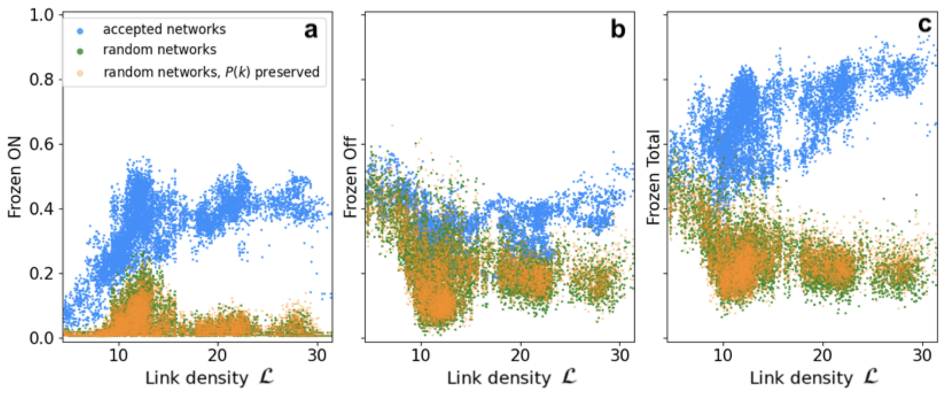

FIGURE 3.17: Frozen components form figure 3.16, before averaging. (**a**) shows only the the portion of nodes that stayed *ON* when tested with 30 random input states. (**b**) shows the share of nodes that stayed *OFF*, and (**c**) shows the total frozen component. Frozen components of accepted CBN mutants are in blue. Frozen components of random networks with the same N, L and degree distribution $P(k)$ as the evolved CBNs are shown in yellow. In green frozen component of random networks with random degree distribution.
Frozen-ON component of this evolved network is considerably larger than the frozen-ON component of the random networks. Frozen component of the evolved network grows with link density $\mathcal{L}$, and frozen components of both types of random network do not.

Figure 3.17 shows the original, un-averaged frozen component values of CBN whose average frozen component is plotted on figure 3.16. We see that the frozen-ON component of random networks is comprised of less than 10% of its nodes, for most values of link density $\mathcal{L}$, but for evolved networks, the frozen-ON component contains in excess of 40% of all nodes. This parallels the observed activation of large regions of *S. cerevisiae* in response to nearly all stimuli [Axelsen, Bernhardsson, and Sneppen, 2008], as can be seen Figure 3.15 (reproduced from [Axelsen, Bernhardsson, and Sneppen, 2008]).

Now, let's consider the frozen component of Canalizing Boolean network evolved with the same link-to-node mutation acceptance ratio as the network from Figures 3.16 and 3.17 above, but with negative net link acceptance probability.

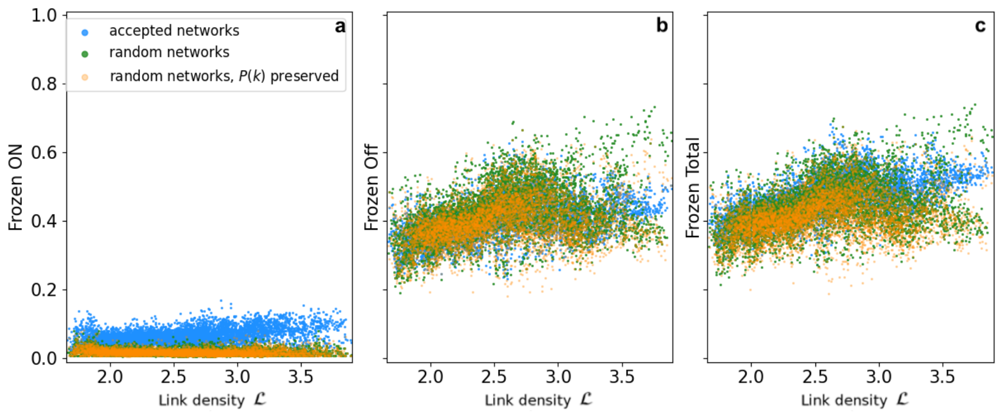

FIGURE 3.18: Frozen ON (**a**), frozen OFF (**b**) and total frozen component (**c**) of CBN evolved with $v \approx 4$ and negative net mutation acceptance probability $\gamma_+ - \gamma_- < 0$. Frozen components of accepted CBN mutants are in blue, and for the random networks: in yellow – for those with the same N, L and degree distribution $P(k)$ as the corresponding CBNs, in green – for those with only the same L and N. This evolved CBN frozen component $\approx$ the randomized networks frozen component

On Figure 3.18 we observe that when $\gamma_+ - \gamma_- < 0$, the frozen component of evolved Canalizing Boolean networks with $v \approx 4$ (same link-to-node mutation acceptance ratio as for the network in Figure 3.17) nearly equals the frozen components of the random networks. Note also the relatively low link density $1.5 \leq \mathcal{L} \leq 4$ this network has.

Let's further explore how the parameters of Canalizing Boolean network evolution affect its frozen component. Total frozen component was measured for accepted mutants of 24 Canalizing Boolean network evolutions and divided by the frozen component of their corresponding randomized networks. The resulting value, that describes how the frozen component of the evolved network compares to that of the frozen component of random corresponding networks, on average, during the entire evolution period, was plotted along the z-axis on Figure 3.19, with link-to-node mutation acceptance ratio along the y-axis, and conditional probability of link loss, p_{ll} along the x-axis. $p_{ll} = 0.5$ corresponds to negative net link addition acceptance, $p_{ll} = 0.4$ means $\gamma_+ = \gamma_-$, and link mutations on average add links for $p_{ll} = 0.3$.

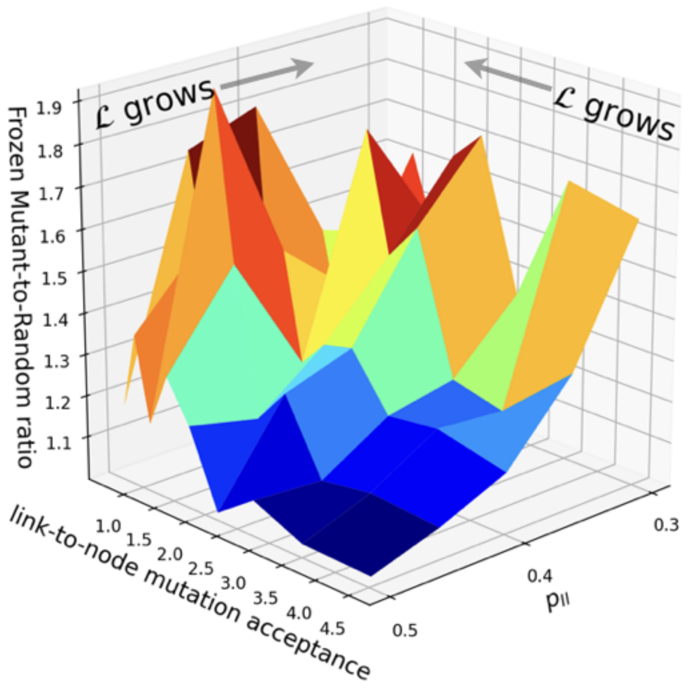

FIGURE 3.19: Total frozen component was measured for accepted mutants of 24 Canalizing Boolean network evolutions, divided by the frozen component of their corresponding randomized networks, and plotted along the z-axis, with CBN link-to-node mutation acceptance ratio v along the y-axis and probability of link loss p_{ll} along x-axis. $p_{ll} = 0.5$ corresponds to negative net link addition acceptance, $p_{ll} = 0.4$ means $\gamma_+ = \gamma_-$, and link mutations on average add links for $p_{ll} = 0.3$. Frozen components for evolved networks with higher link density $\mathcal{L}$ are larger than frozen components for random networks.

We observe that the average frozen component of evolved CBN is considerably larger than frozen components of corresponding randomized networks for smaller v and for non-negative average link acceptance probability. Recalling that average link density $\mathcal{L}$ also tends to be higher for smaller v and for $\gamma_+ - \gamma_- \geq 0$, we plot the same average frozen component ratio (CBN mutant frozen component divided by frozen component of random network) along the y-axis, with CBN average link density $\mathcal{L} = L/N$ along the x-axis on Figure 3.20. We observe that as average link density of evolved Canalizing Boolean networks grows, so does their average frozen component, as compared to the frozen components of the corresponding random networks.

Recall that we have observed earlier, on Figure 3.16 that while the frozen component of evolved CBN was higher when link density was higher, the frozen components of random networks remained approximately 20% for $10 \leq \mathcal{L} \leq 30$. Similarly, on 3.18 the frozen components of random networks remain close to constant (containing approximately 40% of all nodes).

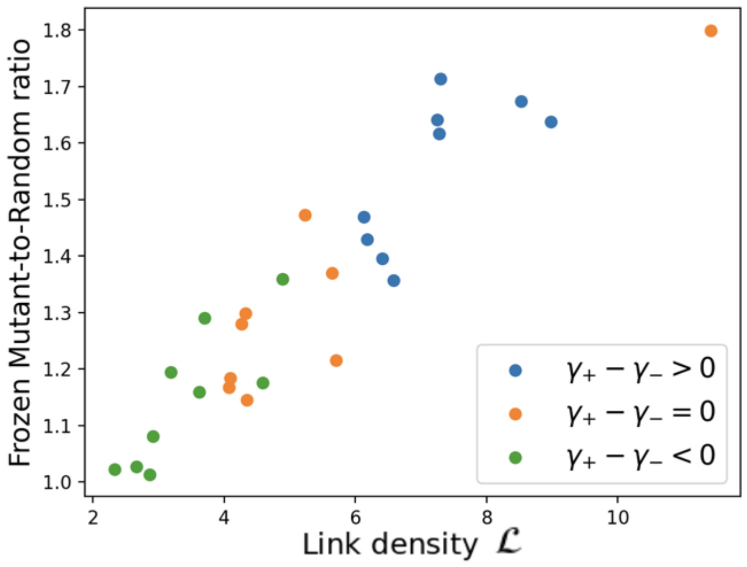

FIGURE 3.20: Total frozen component was measured for accepted mutants of 24 Canalizing Boolean network evolutions, and divided by the frozen component of their corresponding randomized networks. The resulting value describes how average frozen component of the evolved network compares to the average frozen component of the corresponding random networks, and was plotted along the y-axis, with CBN average link density $\mathcal{L}$ along the x-axis. As link density grows so does the average frozen component of the evolved CBN, as compared to the frozen components of the corresponding random networks.

Canalizing Boolean networks evolved in this project were observed to develop large frozen components, with as many as 80% of all nodes not switching during 30 mutation cycles with different random initial conditions. The more connected the evolved

CBN, the larger was its frozen component. For random networks, no increase of frozen component with growing link density was seen. In terms regulatory function, when the frozen component is large, only a small portion of the network nodes participate in signalling, by switching their expression state in response to external/internal stimuli. [Bornholdt and Sneppen, 1998] observed that Boolean networks evolved with the idea that the expression pattern is step-wise preserved exhibit a much more simple expression pattern than random networks with the same link density.

Let us now discuss the frozen component of the randomized networks in more detail. Recall, that two random networks with the same number of links L and number of nodes N as the accepted CBN mutant were generated for each frozen component calculation, one with the degree distribution $P(k)$ equal to that of the accepted mutant, the other – with links distributed randomly among the nodes. In Figures 3.17 and 3.18 the number of nodes that did not switch their state during 30 mutation cycles with random input conditions remained nearly identical for both types of randomized networks.

We observe that the networks evolved with an evolution model that step-wise preserves the expression pattern and traces regulatory signal along all three-node long paths are more prone to developing large frozen components then random Boolean networks with the same number of links, number of nodes, and same degree distribution as the evolved ones.

3.5 Summary

We studied structural properties and expression pattern of Boolean networks evolved using the idea that the expression pattern is step-wise preserved during the evolutionary process as proposed by Bornholdt and Sneppen in 1998 [Bornholdt and Sneppen, 1998], with addition of logical signal tracing along all three-node long paths.

The evolved networks were observed to have predictable structure with exponential-like degree distribution, meanings the degree distribution is wider than the expected

Poisson distribution for random networks. Studies of real transcription factor in-degree distribution show exponential characteristics. The out-degree observed in real genetic regulatory network are wider than exponential, but we believe this is associated with non-transcription factor interactions.

The evolved networks with higher average link density had large frozen-ON components, also similar to observations of real transcription networks of *Saccharomyces cerevisiae*.

Randomized networks with the same number of nodes, same number of links, and the preserved degree distribution of the evolved networks, had the same frozen components as the fully randomized networks. Both of types of random networks have considerably smaller frozen components than the corresponding evolved networks. Thus our stepwise evolution favors large frozen components, in particular when connectivity is high.

Appendix A

Simulation Code

(Python 3.8.2 was used with numpy)

```
import numpy as np

r = np.random

def add_link(mtx, ands, f_into, f_from):
    m = np.copy(mtx)
    a = np.copy(ands)
    m[f_into, f_from] = np.amax(m[f_into, :]) + 1
    ra = r.randint(3)
    if ra > 1:
        a[f_into, f_from] = -1
    elif ra == 1:
        a[f_into, f_from] = 1
    return m, a

def remove_link(mtx, ands, l_into, l_from):
    m = np.copy(mtx)
    a = np.copy(ands)
    m[l_into, m[l_into, :] > m[l_into, l_from] ] -= 1   #reduce by 1 the rank
    m[l_into, l_from] = 0
    a[l_into, l_from] = 0
    return m, a


def lose_node(mtx, inx):
    m = np.copy(mtx)                                    # copy here is needed,
    links_from_inx = np.nonzero(m[:, inx])[0]           # nodes that receive li
    if np.size(links_from_inx) > 0:
        for n in links_from_inx:                        # adjust link ranks for
            m[n, m[n, :] > m[n, inx] ] -=1
    for axis in range(2):
        m = np.delete(m, inx, axis)
    return m


def expand_mtx(mx, inx):
    N = np.shape(mx)[0]
    m = np.zeros((N+1, N+1))
    m[N, 0:N] = np.copy(mx[inx, :])
    m[0:N, 0:N] = np.copy(mx)
    return m


#removed: self link of original lead to links from duplicate to original
def dupl_node(mtx, inx):
    N = np.shape(mtx)[0]
    m = expand_mtx(mtx, inx)                          # add node to the network
    maxs = np.amax(m, axis = 1 ) + 1
    links_from_inx = np.nonzero(m[:, inx])[0]         # nodes that receive links
    m[links_from_inx, N] = maxs[links_from_inx]       # assign link 'order rank'
    if m[inx, N] != 0:                                # Link from duplicate to o
        m[N, N] = m[inx, N]                           # link duplicate to itself
        m[inx, N] = 0                                 # unlink originl from dupl
    return m
```

```
def link_type_link(mtx, p_lose_link):
    if r.random() < p_lose_link:
        link_in, link_out = np.nonzero(mtx)
        if np.size(link_in) > 0:
            n = r.randint(len(link_in))
            return 1, link_in[n], link_out[n]
        return 0, 0, 0
    free_in, free_out = np.where(mtx==0)
    if np.size(free_in) > 0:
        n = r.randint(len(free_in))
        return 2, free_in[n], free_out[n]
    return 0, 0, 0


def update(mtx, ands, state_old):
    state = np.copy(state_old)                          #corrected form some crazy
    if np.any(mtx == 1):
        ranks = np.arange(1, np.amax(mtx)+1)            #corrected from np.ara
        for rank in ranks:                              #loop through all link
            links = np.where( mtx == rank )             #'coordinates' of link
            for into, frm in zip( links[0], links[1] ):
                if ands[into, frm] == -1:
                    state[into] = 0
                elif ands[into, frm] == 1:
                    state[into] *= state_old[frm]       #if rank == 1: initial
                else:
                    state[into] += state_old[frm]       #if rank == 1 initial
        state[state > 0] = 1
    return state
```

```
def init_mut(m, a, n_steps):
    m_st = r.randint(2, size = np.shape(a)[0])
    d_st = np.copy(m_st)
    link_in, link_out = np.where(m==0)
    n = r.randint(len(link_in))
    d_m, d_a = add_link(m, a, link_in[n], link_out[n])
    for i in range(n_steps):
        d_st_new = update(d_m, d_a, d_st)
        m_st_new = update(m, a, m_st)
        if np.any(m_st_new - d_st_new):
            return m, a
        d_st = d_st_new
        m_st = m_st_new
    return d_m, d_a


def init_ntwrk(k, N, nst):
    m = np.zeros((N, N))                                    #network matrix
    a = np.zeros((N, N))                                    #and matrix
    conn = 0
    while conn < k:
        m, a = init_mut(m, a, nst)
        conn = np.count_nonzero(m)/float(N)
    return m, a


def unlinked(m):
    nodes = np.arange( np.shape(m)[0] )
    return np.intersect1d(nodes[np.all(m == 0, axis = 1)], nodes[np.all(m =
def fill_states(mtx, ands, inpt, n):
    states = np.zeros( (n, len(inpt) ))
    states[0, :] = update(mtx, ands, inpt)
    for i in range(1, n):
        states[i, :] = update(mtx, ands, states[i-1, :])
    return states


def frozens(mx, ax, n):
    n_inpts = 40
    N = np.shape(mx)[0]
    states = np.zeros((n_inpts*n, N))
    for i in range(n_inpts):
        inpt = r.randint(2, size = N)
        states[ i*n : (( 1+i)*n), :] = fill_states(mx, ax, inpt, n)

    f = np.sum(states, axis = 0)
    return np.count_nonzero(f == n*n_inpts), np.count_nonzero(f == 0)


def link_mut_frz(mtx, ands, p_lose_link, n_steps):
    type_link, sink, source = link_type_link(mtx, p_lose_link)
    if type_link == 0:
        return mtx, ands, [-1, -1]
    if type_link == 1:
        d_mtx, d_ands = remove_link(mtx, ands, sink, source)
    elif type_link == 2:
        d_mtx, d_ands = add_link(mtx, ands, sink, source)
    N = np.shape(ands)[0]
    m_state = r.randint(2, size = N)
    d_state = np.copy(m_state)
    for i in range(n_steps):
        m_state_n = update(mtx, ands, m_state)
        d_state_n = update(d_mtx, d_ands, d_state)
        if np.any(m_state_n - d_state_n):
            return mtx, ands, [-1, -1]
        m_state = m_state_n
        d_state = d_state_n
    un_linked = unlinked(d_mtx)
    if np.size(un_linked) > 0:
        for a in range(2):
            d_mtx = np.delete(d_mtx, un_linked, axis = a)
            d_ands = np.delete(d_ands, un_linked, axis = a)
        return d_mtx, d_ands, [-3, -3]
    return d_mtx, d_ands, frozens(d_mtx, d_ands, n_steps)


def lose_mut_frz(mtx, ands, n_steps, N_min):
    N = np.shape(ands)[0]
    if N < N_min:
        return mtx, ands, [-1, -1]
    node = r.randint(N)
    d_mtx = lose_node(mtx, node)
    d_ands = np.copy(ands)
    for axis in range(2):
        d_ands = np.delete(d_ands, node, axis)
    m_state = r.randint(2, size = N)
    d_state = np.copy(m_state)
    d_state = np.delete(d_state, node)
    for i in range(n_steps):
        m_state_n = update(mtx, ands, m_state)
        d_state_n = update(d_mtx, d_ands, d_state)
        if np.any(np.delete(m_state_n, node) - d_state_n):
            return mtx, ands, [-1, -1]
        m_state = m_state_n
        d_state = d_state_n
    un_linked = unlinked(d_mtx)
    if np.size(un_linked) > 0:
        for a in range(2):
            d_mtx = np.delete(d_mtx, un_linked, axis = a)
            d_ands = np.delete(d_ands, un_linked, axis = a)
        return d_mtx, d_ands, [-3, -3]
    return d_mtx, d_ands, frozens(d_mtx, d_ands, n_steps)


def dupl_mut_frz(mtx, ands, n_steps):
    N = np.shape(ands)[0]
    i_state = r.randint(2, size = (N+1))
    node = r.randint(N)
    d_mtx = dupl_node(mtx, node)
    d_ands = expand_mtx(ands, node)
    d_ands[0:N, N] = np.copy(d_ands[0:N, node])
    if d_ands[node, N] != 0:                        #if a link from doupli
        d_ands[N, N] = d_ands[node, N]              #self-link the duplica
        d_ands[node, N] = 0                          #delelte link from du
    d_state = r.randint(2, size = (N+1))
    m_state = np.copy(d_state[0:N])
    for i in range(n_steps):
        m_state_n = update(mtx, ands, m_state)
        d_state_n = update(d_mtx, d_ands, d_state)
        if np.any(d_state_n[0:N] - m_state_n):
            return mtx, ands, [-3, -3]
        m_state = m_state_n
        d_state = d_state_n
    return d_mtx, d_ands, frozens(d_mtx, d_ands, n_steps)
```

```
def link_mutation(mtx, ands, p_lose_link, n_steps):
    type_link, sink, source = link_type_link(mtx, p_lose_link)
    if type_link == 0:
        return mtx, ands
    if type_link == 1:
        d_mtx, d_ands = remove_link(mtx, ands, sink, source)
    elif type_link == 2:
        d_mtx, d_ands = add_link(mtx, ands, sink, source)
    N = np.shape(ands)[0]
    m_state = r.randint(2, size = N)
    d_state = np.copy(m_state)
    for i in range(n_steps):
        m_state_n = update(mtx, ands, m_state)
        d_state_n = update(d_mtx, d_ands, d_state)
        if np.any(m_state_n - d_state_n):
            return mtx, ands
        m_state = m_state_n
        d_state = d_state_n
    un_linked = unlinked(d_mtx)
    if np.size(un_linked) > 0:
        for a in range(2):
            d_mtx = np.delete(d_mtx, un_linked, axis = a)
            d_ands = np.delete(d_ands, un_linked, axis = a)
        return d_mtx, d_ands
    return d_mtx, d_ands


def lose_mutation(mtx, ands, n_steps, N_min):
    N = np.shape(ands)[0]
    if N < N_min:
        return mtx, ands
    node = r.randint(N)
    d_mtx = lose_node(mtx, node)
    d_ands = np.copy(ands)
    for axis in range(2):
        d_ands = np.delete(d_ands, node, axis)
    m_state = r.randint(2, size = N)
    d_state = np.copy(m_state)
    d_state = np.delete(d_state, node)
    for i in range(n_steps):
        m_state_n = update(mtx, ands, m_state)
        d_state_n = update(d_mtx, d_ands, d_state)
        if np.any(np.delete(m_state_n, node) - d_state_n):
            return mtx, ands
        m_state = m_state_n
        d_state = d_state_n
    un_linked = unlinked(d_mtx)
    if np.size(un_linked) > 0:
        for a in range(2):
            d_mtx = np.delete(d_mtx, un_linked, axis = a)
            d_ands = np.delete(d_ands, un_linked, axis = a)
        return d_mtx, d_ands
    return d_mtx, d_ands

def dupl_mutation(mtx, ands, n_steps):
    N = np.shape(ands)[0]
    node = r.randint(N)
    d_mtx = dupl_node(mtx, node)
    d_ands = expand_mtx(ands, node)
    d_ands[0:N, N] = np.copy(d_ands[0:N, node])
    if d_ands[node, N] != 0:                              #if a link from douplica
        d_ands[N, N] = d_ands[node, N]                    #self-link the duplicate
        d_ands[node, N] = 0                            #delelte link from d
    d_state = r.randint(2, size = (N+1))
    m_state = np.copy(d_state[0:N])
    for i in range(n_steps):
        m_state_n = update(mtx, ands, m_state)
        d_state_n = update(d_mtx, d_ands, d_state)
        if np.any(d_state_n[0:N] - m_state_n):
            return mtx, ands
        m_state = m_state_n
        d_state = d_state_n
    return d_mtx, d_ands



def order_inputs(mtx):
    sinks, sources = np.nonzero(mtx)
    for sink in np.unique(sinks):
        ordered_vals = np.arange(np.count_nonzero(sinks == sink))+1
        r.shuffle(ordered_vals)
        mtx[sink, sources[sinks==sink]] = ordered_vals
    return mtx



def randomize_smpl(ntwrk, ands_mt, num_steps):       #randomize network with
    ntw = np.copy(ntwrk)
    ands = np.copy(ands_mt)
    N = np.shape(ntw)[0]
    z0, z1 = np.nonzero(ntw)                         # coordinates of nonzer
    ntw = ntw.flatten()
    r.shuffle(ntw)
    ntw = ntw.reshape(N, N)
    ntw = order_inputs(ntw)
    ands_values = ands[z0, z1]                    # values of ands for nonze
    r.shuffle(ands_values)
    ands[z0, z1] = 0                              # put current ands values
    ands[np.nonzero(ntw)] = ands_values           # assign shuffled ands val

    return frozens(ntw, ands, num_steps)



def randomize_presrv_degres(ntw, ands, num_steps):       #randomize network pr
    m = np.copy(ntw)
    matr_a = np.copy(ands)
    N = np.shape(ntw)[0]
    count = 0
    while count < (int(1.5*np.count_nonzero(ntw))):
        sources = np.unique(np.nonzero(m)[1])
        a = r.choice(sources)
#randomly choose source A
        sinks_a = np.unique(np.nonzero(m[:, a]))
#sinks that A links to
        sinks_a = sinks_a[sinks_a != a]
#remove A from sinks
        if np.size(sinks_a) > 0:
            b = r.choice(sinks_a)
# choose B from sinks (doesn't contain A)
            sources = sources[ (sources != a) & (sources != b ) ]
#delete B and A from sources
            sources = np.setdiff1d(sources, np.nonzero(m[b, :]))
#delete sources that link into B ( Sources that link into A are fine)
            if np.size(sources) > 0:
                c = r.choice(sources)
#chose a source C that does'nt link to B and isn't A or B
```

```
                sinks = np.unique(np.nonzero(m[:, c]))
#all skinks from C
                sinks = np.setdiff1d(sinks, np.nonzero(m[:, a]))
#delete sinks that A links into (d may not be linked with A, but may link
                if np.size(sinks) >0:
                    d = r.choice(sinks)
#from allowed sinks (not linked or = to A or B) select D
                    m[(b, d, d, b), (a, c, a, c)] = np.array([0, 0, 1, 1])
#switch A-->B, C-->D to A-->D and C-->B
                    matr_a[(d, b), (a, c)] = matr_a[(b, d), (a, c)]
#adjust ands matrix to preserve protein type
                    matr_a[(b, d), (a, c)] = 0
                    count +=1
    m = order_inputs(m)
    return frozens(m, matr_a, num_steps)

def prep_mutation(ntw, ands, n_steps, N_min, N_max, p_lilo, p_li):
    if r.random() < p_li:
        return link_mutation(ntw, ands, p_lilo, n_steps)
    N = np.shape(ands_network)[0]
    if r.random() < N/2.0/float(N_max):
        return lose_mutation(ntw, ands, n_steps, N_min)
    else:
        return dupl_mutation(ntw, ands, n_steps)



def boundary_mutation(ntw, ands, n_steps, N_min, N_max, p_lilo, p_li):
    if r.random() < p_li:
        return link_mut_frz(ntw, ands, p_lilo, n_steps)
    N = np.shape(ands_network)[0]
    if r.random() < N/2.0/float(N_max):
        return lose_mut_frz(ntw, ands, n_steps, N_min)
    else:
        return dupl_mut_frz(ntw, ands, n_steps)


def steady_state(ntw, ands, n_steps, N_min, N_max, p_lilo, p_li):
    result = np.zeros(8)
    ntw, ands, result[6:] = boundary_mutation(ntw, ands, n_steps, N_min, N_max
    #frozen component computed from network (on, off)
    netw_size = np.shape(ntw)[0]
    result[0] = netw_size
    result[1] = np.count_nonzero(ntw)
#total number of links in network
    if np.any(result[6:] < 0):
        result[2:] = -3
        return ntw, ands, result

    result[2:4] = randomize_presrv_degres(ntw, ands, n_steps)
#frozen of, frozen off for network randomized with preserved degree distributi
    result[4:6] = randomize_smpl(ntw, ands, n_steps)
#frozen on, frozen off, for simply randomized network

    return ntw, ands, result




#initial values
p_links =           #probability to attempt link mutation
p_linkloss =        #conditional probability to chose link loss in case oflink
depth =             #number of nodes to trace signal paths between
N_start =           #initial size of network
k_start =           #initial connectivity
```

```
N_mutations =        #number of mutations to attempt
N_max =              #weak network size constraint
N_min =              #minimum network size



network, ands_network =  init_ntwrk(k_start, N_start, depth)
un_linked = unlinked(network)
for axis in range(2):
    network = np.delete(network, un_linked, axis)
    ands_network = np.delete(ands_network, un_linked, axis)
print('aft_del_unl', np.shape(network))

netw_size = N_start
temp = np.zeros(2)

while (netw_size<N_max):            #grow the network to steady state
    if r.random() < p_links:
        network, ands_network =  link_mutation(network, ands_network, p_lin
    elif r.random() < 0.5:
        network, ands_network = lose_mutation(network, ands_network, depth,
    else:
        network, ands_network = dupl_mutation(network, ands_network, depth)
    netw_size = np.shape(network)[0]

data = np.zeros((N_mutations, 8))

for i in range(N_mutations):            #stationary state evolution here
    network, ands_network, data[i, :] = steady_state(network, ands_network,
```

Bibliography

Axelsen, Jacob, Sandeep Krishna, and Kim Sneppen (Dec. 2007). "Cost and Capacity of Signaling in the Escherichia coli Protein Reaction Network". In: *Journal of Statistical Mechanics Theory and Experiment* 2008. DOI: 10.1088/1742-5468/2008/01/P01018.

Axelsen, Jacob Bock, Sebastian Bernhardsson, and Kim Sneppen (Mar. 2008). "One hub-one process: a tool based view on regulatory network topology". English. In: *B M C Systems Biology* 2, p. 25. ISSN: 1752-0509. DOI: 10.1186/1752-0509-2-25.

Bornholdt, Stefan and Kim Sneppen (1998). "Neutral Mutations and Punctuated Equilibrium in Evolving Genetic Networks". In: *Phys. Rev. Lett.* 81 (1), pp. 236–239. DOI: 10.1103/PhysRevLett.81.236.

Erdös, P. and A Rény (1960). "On the evolution of random graphs". In: *Publications of the Mathematical Institute of the Hungarian Academy of Sciences* 5 (1), pp. 17–60. DOI: 10.1103/PhysRevLett.81.236.

Kauffman, Stuart A. (1990). "Requirements for evolvability in complex systems: Orderly dynamics and frozen components". In: *Physica D: Nonlinear Phenomena* 42.1, pp. 135–152. ISSN: 0167-2789. DOI: https://doi.org/10.1016/0167-2789(90)90071-V. URL: https://www.sciencedirect.com/science/article/pii/016727899090071V.

Lee, Tong Ihn et al. (2002). "Transcriptional Regulatory Networks in Saccharomyces cerevisiae". In: *Science* 298.5594, pp. 799–804. ISSN: 0036-8075. DOI: 10.1126/science.1075090. URL: https://science.sciencemag.org/content/298/5594/799.

Maslov, Sergei, Kim Sneppen, and Alexei Zaliznyak (2004). "Detection of topological patterns in complex networks: correlation profile of the internet". In: *Physica A: Statistical Mechanics and its Applications* 333, pp. 529–540. ISSN: 0378-4371.

DOI: https://doi.org/10.1016/j.physa.2003.06.002. URL: https://www.sciencedirect.com/science/article/pii/S0378437103008409.

"Traffic and Queueing Theory" (2008). In: *Mathematics for Engineers*. John Wiley Sons, Ltd. Chap. 6, pp. 289–361. ISBN: 9780470611449. DOI: https://doi.org/10.1002/9780470611449.ch6. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9780470611449.ch6. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470611449.ch6.

Trusina A Sneppen K, Dodd IB Shearwin KE Egan JB (2005). "Functional Alignment of Regulatory Networks: A Study of Temperate Phages". English. In: *PLoS Comput Biol*. DOI: https://doi.org/10.1371/journal.pcbi.0010074.