Product diversity and network structure: A minimal model

Gia-Huy Dang

supervised by

Prof. Marc-Thorsten Hütt
Prof. Julia Bendul

Technical Report
July 2017

This project is funded by the gLINK program

Product diversity and network structure: A minimal model

Gia-Huy Dang

Abstract

Synchronization of oscillators is a widely-known phenomenon that has presented and been studied in various dynamical networks, ranging from social to biological and technical systems. The most successful and widely-used tool for measuring synchronization in such complex systems is Kuramoto model. In this work, we study a minimal model of evolved flow distribution networks. Our main goal is to observe how the networks synchronize, and the relationship between Kuramoto's order parameter and the diversity of the networks' prescribed output pattern. We found a relationship between the number of links in one graph and its synchronization performance, as well as which property of evolved flow networks decides the synchronization scheme.

Contents

1. Introduction

Network thinking is becoming an essential to many branches of sciences as many systems of scientific interest can be represented as networks for studying [1]. Examples comes from many fields: World Wide Web, food web, financial web, social net, neural network, transportation network, manufacturing network (Need citations). In order to understand the functions of real-world network systems, studying about the impact of complex network topology on dynamical systems is necessary [1, 2, 3].

Synchronization on complex networks has been studied intensively as an universal concept in many fields of study, especially in neural science [4, 5]. There has been many previous studies focus on synchronization of small-world networks, or scale-free networks. Our work consider flow distribution networks with different structure, which represent the simple manufacturing system in industrial production.

In manufacturing systems, there exists two types of phenomena for synchronization: logistics and physics [7]. This study based on previous work on evolved flow networks by Beber et al.(2013)[6], which has shown that the network architectures are strongly affected by the prescribed output pattern. We want to investigate about intrinsic physics synchronization on evolved flow networks. A relationship between the the diversity of prescribed output pattern and the order parameter of synchronization for a given network should be found; we also curious about whether the robustness of a network relates to its synchronization. For this purpose, we first derive the backgrounds of flow distribution network, evolutionary network algorithm and Kuramoto model (section 2), then introduce a minimal network construction for studying (section 3) and the show results of experiment (section 4).

2. Backgrounds

2.1 Flow Distribution Network

We first summarize the flow distribution network model which is introduced in [8, 9, 10]. The network G consists of 3 groups: The input layer N_{in}, the middle layer with size $N_{mid}$and the output layer with N_{out}. Directed connections come from the input layer to the middle layer, and from the middle layer to output layer. The network G is defined by the adjacency matrix A, which represents each directed connection from $j \to i$ as $A_{ij} = 1$. A unit flux is applied to each input node.The incoming flux at each node is equally distributed through the node's outgoing links, and is gathered at the output node. The total flux at node i is computed as in reference [8]:

$$x_i = \sum_{k=1}^{N} \frac{a_{ik}}{\sum_{l=1}^{N} a_{lk}} x_k. \tag{2.1}$$

where x_i is the flux at node i and a_{ik} is the weight from node k to node i. In the case of a binary adjacency (i.e. $A_{ij} = 1$ if $j \to i$ and 0 otherwise), the term $\frac{a_{ik}}{\sum_{l=1}^{N} a_{lk}}$ becomes one over the out-degree of node k. The flux x_i can be found by solving the set of equation (1), for i= 1, 2, ..., N. Note that $x_i = 1$ for all the input nodes.
For a given unit flux is applied at node α, denoted by $x_\alpha = 1$, it is distributed through directed links in the network and reach various output nodes. We denote $Q_{\beta\alpha}$ as the fractions of the unit flux from input node α to output node β. The matrix Q with elements $Q_{\beta\alpha}$ and size N_{out} x N_{in} is the output pattern of the network G. Notice that the total flow is conserved in the network, or $\sum_{\beta=1}^{N_{out}} Q_{\beta\alpha} = 1$, for any α. An example of flow distribution sub-network at a given input node A is illustrated in Fig 2.1a, and a graphic visualization of an output pattern in Fig 2.1b.

For each network, a random prescribed output pattern Q^0 is created, specifying to which final destinations (the output nodes) and in what amounts (the proportion) a particular unit flux from one input node should be supplied. The distance - or the flow error - ϵ between the output pattern and the prescribed one is defined as

$$\epsilon = |Q - Q^0| = \frac{1}{2N_{in}} \sum_{\alpha=1}^{N_{in}} \sum_{\beta=1}^{N_{out}} (Q_{\beta\alpha} - Q^0_{\beta\alpha})^2 \tag{2.2}$$

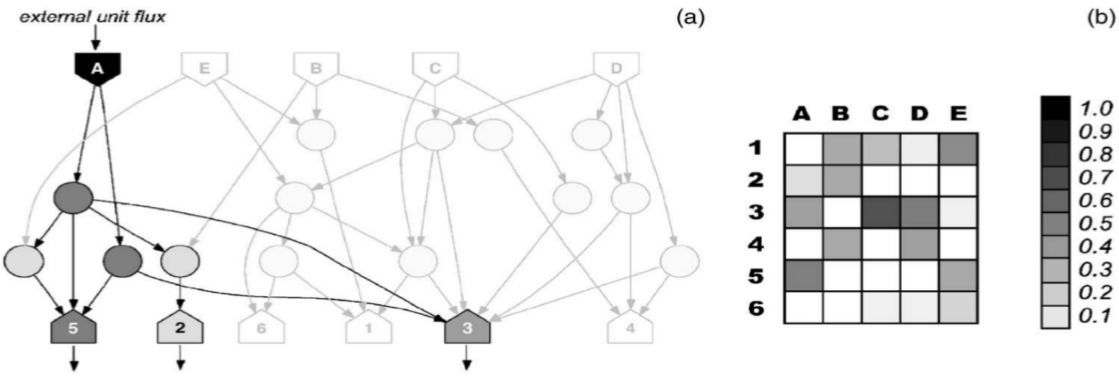

Figure 2.1: Example of (a) flow distribution network and (b) prescribed output pattern

Another characteristic that we want our network to have, is the robustness to local damage, such as losing a link or a node, during the process. A network is robust with respect to a particular performance if its performance does not deteriorate much under damage [10]. To measure functional robustness of a network G, a set S which contains G applied to local damage is considered. A threshold h is introduced, such that all networks in S that have $\epsilon > h$ are abortive. The robustness ρ of G is defined as

$$\rho = \frac{1}{N_S} \sum_{i=1}^{N_S} \theta(h - \epsilon_i) \tag{2.3}$$

where N_S is number of networks in S and θ is the step function:

$$\theta(z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$$

The work to construct a network with prescribed output pattern which has small flow error and is robust against local damages can be seen as an optimization problem. The tasks are trying to minimize the flow errors, while maximize the robustness (as possible). In [6], the network is optimized - or 'evolved' - through two phases: The first phase is trying to get the network's flow error ϵ to become lower than the threshold h by using a variant of simulated annealing. The following optimization algorithm is implemented, as in [10]. Each iteration consists of same steps

1. Take network G and calculate flow error ϵ.
2. Apply path mutation scheme to G, obtain G' and calculate $epsilon'$.

3. Calculate $\Delta\epsilon = \epsilon' - \epsilon$.
4. If $\Delta\epsilon < 0$, we accept the mutation, replacing G by G' and ϵ by ϵ'; if $\epsilon > 0$, we accept the mutation with probability $exp(-\Delta\epsilon/\epsilon\sigma)$.
5. Return to step 1.

Figure 2.2 shows how flow errors in the first phase for a given network.

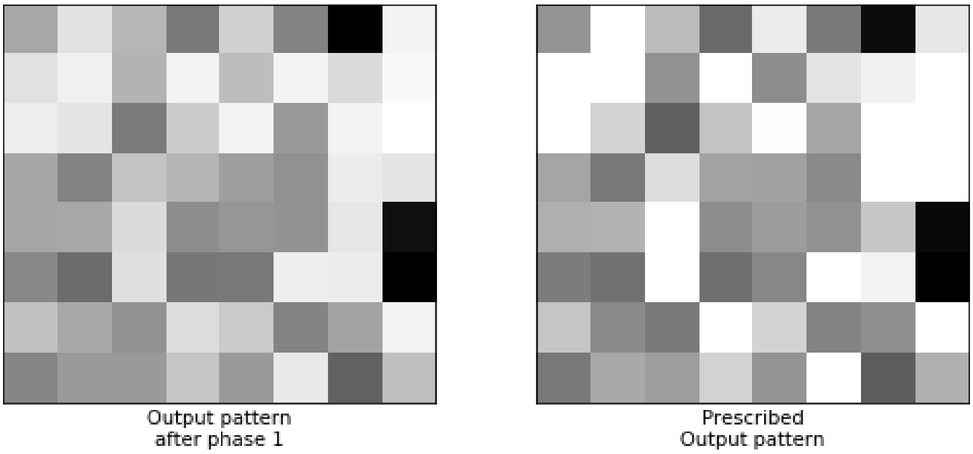

Figure 2.2: The output pattern after finishing phase 1 compared to the prescribed output pattern

Once the network achieves ϵ which is below the threshold h, the second phase of the algorithm is introduced.. This first phase is called "pattern recognition" in [6].

In the second phase of the algorithm, the network structure is altered to maintain its flow error below the threshold h under local damages. The idea arises from observations of biological cells. People wondered how a genetic network performs if one gene is knocked out, or what happens with an animal if one of its synaptic connections is broken. The answer is quite surprising: In most cases, noting dramatic would happen. The gene network or animal will keep functioning in the same way, ignoring the damage. It is because the biological systems are designed in such a way that it can achieve high robustness. If a link or a node in the network is destroyed, signal will pass to its destination through another route. Networks that can do this is called 'self-correcting' networks.

The process is almost the same with the first phase:

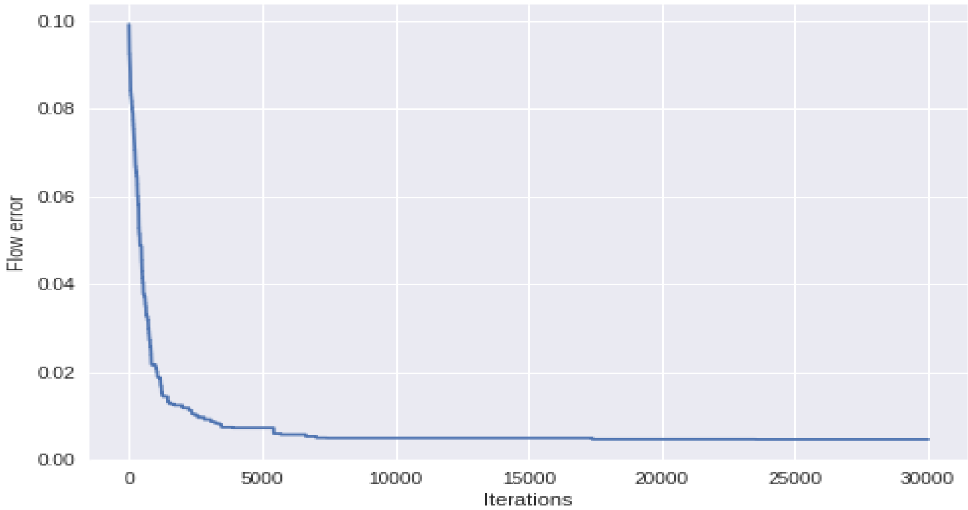

Figure 2.3: Evolution of flow error ϵ with parameters $N_{in} = 8$, $N_{middle} = 8$, $N_{out} = 8$, $K = 4$, $h = 0.007$, $\sigma = 10^{-4}$

1. Take G and calculate ϵ and its robustness ρ.
2. Obtain G' through path mutation, calculate ϵ' and ρ'.
3. Calculate $\Delta\epsilon = \epsilon' - \epsilon$ and $\Delta\rho = \rho - \rho'$.
4. If the error of G' is larger than h, the decision is based on flow errors. Here, it is the same with the first phase: accept the mutation if $\Delta\epsilon < 0$ and accepted with probability $exp(-\Delta\epsilon/\epsilon\sigma)$ if $\Delta\epsilon > 0$. If $\epsilon' < h$, the decision is based on the robustness. We accept the mutation if $\Delta\rho < 0$, and if $\Delta\rho > 0$, it is accepted with probability $exp(-\Delta\rho/(1-\rho)\sigma)$.
5. If the mutation is accepted, G is replaced with G' as well as ϵ and ρ. If not, then G will be used in the next iteration.
6. Return to step 1.

2.2 Synchronization on evolved flow distribution network

In manufacturing system, physics synchcroniztion is defined as "the rhythm and repetitive behaviour of production processes"[7]. We do a small investigation on our small dataset which is generated using evolutonary network, to observe how well the network synchronized, using the Kuramoto model[11]. This is given by:

$$\frac{\partial \theta_i}{\partial t} = \omega_i + \frac{K}{N} \sum_{j=1}^{N} sin(\theta_j - \theta_i), i = 1, ..., N \tag{2.4}$$

where θ_i denotes the phase of the i^{th} oscillator, with its natural frequency ω_i randomly generated from the Lorentzian probability density:

$$g(\omega) = \frac{\gamma}{\pi[\gamma^2 + (\omega - \omega_0)]} \tag{2.5}$$

with width γ and mean ω_0. Kuramoto further introduced the order paramter, which is the result of solving Eq.(4):

$$r(t) = |\frac{1}{N} \sum_{i=1}^{N} e^{i\theta_i(t)}| \tag{2.6}$$

in the limit of $N \to \infty$ and $t \to \infty$.

2.3 Finding strong connections components

A directed graph is called strongly connected if every vertex is reachable from every other vertex. One directed graph can be broken down into strongly connected components, which is its subgraphs that connected strongly. There are several algorithms for finding strongly connected components of a given directed graph in linear time [12, 13, 14]. Despite the fact that Kosaraju's algorithm is conceptually simple [12], Tarzan's algorithm uses depth-first search only once, which is more optimized. Note that from now on, we call strongly connected components as SCC for short.

Tarzan's algorithm are based on the depth-first search. The vertices are indexed as they are traversed by depth-first search procedure. While returning from the recursion of DFS, every vertex V gets assigned a vertex L a a representative. L is a vertex with the least index that can be reach from V. Nodes with the same representative assigned are located in the same SCC.

The steps (pseudocode) of this algorithm is as follow:

Algorithm 1: Tarzan's algorithm for finding strongly connected components in graph G

```
index := 0;
S := empty array;
for v in V do
    if v.index is undefined then
        SCC(v);
    end
end
Function SCC(v)
    v.index := index;
    v.lowlink := index;
    index := index + 1;
    S.push(v);
    v.onStack := True;
    for (v, w) in E do
        if w.index is undefined then
            SCC(w) v.lowlink := min(v.lowlink, w.lowlink)
        else if w.onstack then
            v.lowlink := min(v.lowlink, w.index)
    end
    if v.lowlink = v.index then
        while  w != u do
            w := S.pop();
            w.onStack := False;
            output the current strongly connected component
        end
    end
end
```

3. Documentation of Experiments

We conduct our experiment as follow: Firstly, a random graph represents a simple manufacturing system is generated, which has three layers: N_{in}, N_{middle}, N_{out}. The graph should satisfy the conditions mentioned in section 2.1. Then a corresponding prescribed output pattern is made. Notice that in [8, 9, 10], only $W = 4$ output nodes received fluxes from an activated input node. It is obvious that, the less K is, the more diverse the products are generated. Since we are studying about the product diversity, we consider the case when $W = 2$ and $W = 0$.

We then apply the first phase of the evolution to the graphs. For the second phase, we consider two cases of local damage for the second phase of the evolution: removal of links and removal of nodes. In the case of node robust networks, each node is deleted once; for the case of link robust network, each link is deleted once. For both two cases, the second phase of the evolutionary algorithm stops when G achieves robustness $\rho = 1.0$ or after 3 x 10^5 iterations. Other parameters are also fixed: the threshold h is set at 0.007, and $\sigma = 10^{-4}$. We then select networks with robustness $\rho \geq 0.6$ and achieve our small dataset, which is shown in Table 3.1.

Evolution type	W	Number of graphs
link robust	W = 0	50
	W = 2	40
	W = 4	40
node robust	W = 0	50
	W = 2	40
	W = 4	40

Table 3.1: Type and number of graphs in the dataset

After the evolutionary phase, we measure how the network synchronizes by calculating the order parameter, using Kuramoto model. Since the phase and frequency for each node in the network are randomly generated, for each evolved graph G we run the Kuramoto model 100 times with 1000 time steps per iteration, and achieve the averaged order parameter r.

4. Results

The goal of this project is to seek for relationship between the prescribed output pattern and the synchronization of the network, in the increasing of product diversity.

We first observe the difference in 2 random graphs which have 500 nodes and no specified layers in synchronization. We make one graph 'denser' than the other one, by increasing the number of links between nodes in that graph; here, we use parameter p to describe the ratio of links in a graph, compare to its fully connected version, which has N x $(N-1)$ edges. The result is shown in Figure 4.1. We expect that there is a relationship between the order parameter and the number of links in a graph. To prove it, we first modify equation 2.4:

$$\frac{\partial \theta_i}{\partial t} = \omega_i + \frac{K}{L} \sum_{j=1}^{N} sin(\theta_j - \theta_i), i = 1, ..., N \tag{4.1}$$

by replacing the size N with L, which represents the number of links in a given network.

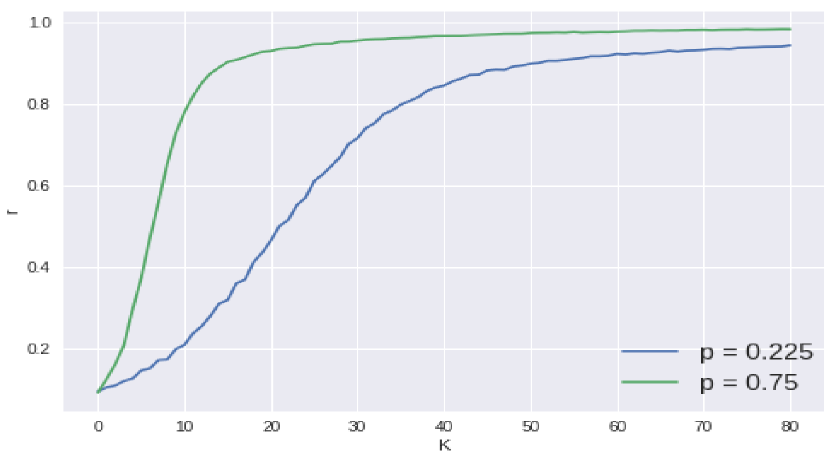

Figure 4.1: The behaviour of order parameter as K increases, for two graphs with different number of links

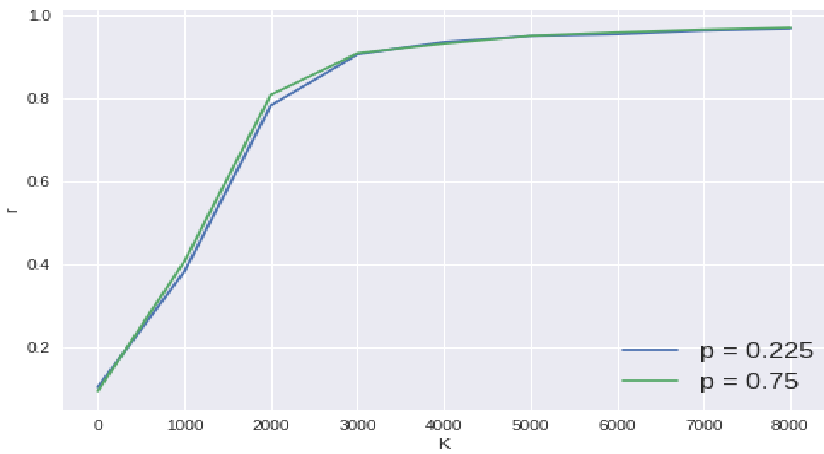

Figure 4.2: The behaviour of order parameter as K increases, for two graphs with different number of links, after normalized

The result in Figure 4.2 is quite satisfying: After normalizing, both graphs are nearly identical, which proves that the number of links in one graph is a factor to decide how that graph synchronizes. Note that K in Figure 4.2 is much larger than coupling strength in Figure 4.1, since replacing N by L makes the constant fraction to become much smaller.

In a given evolved network, we consider each node as an oscillator and observe how they synchronize as the coupling strength K increase. As stated in [1], the larger K is, the higher value in order parameter r that we get, which means more and more oscillators (node) are synchronized. Figure 4.3 illustrates the result when apply Kuramoto model to networks with different level of diversity.

We have figured out that the number of links in one network is related to its synchronization stage. As these three figures show, the order parameters between them after apply $K = 5000$ are not significantly different. However, as W increases, graphs evolve from the link robust phase seem to become more dense than graphs evolve from node robust phase, which also results in higher r. However, the order parameter r is nowhere close to 1.0, for all cases. This can be explained if we take a closer look to the structure of our evolved networks: The input layer and output layer, compared to middle layer, are more simple and directed. To be specific, the input nodes take no interact with each other and so do the output nodes, whereas in the middle layer, nodes have some interactions between each other. These

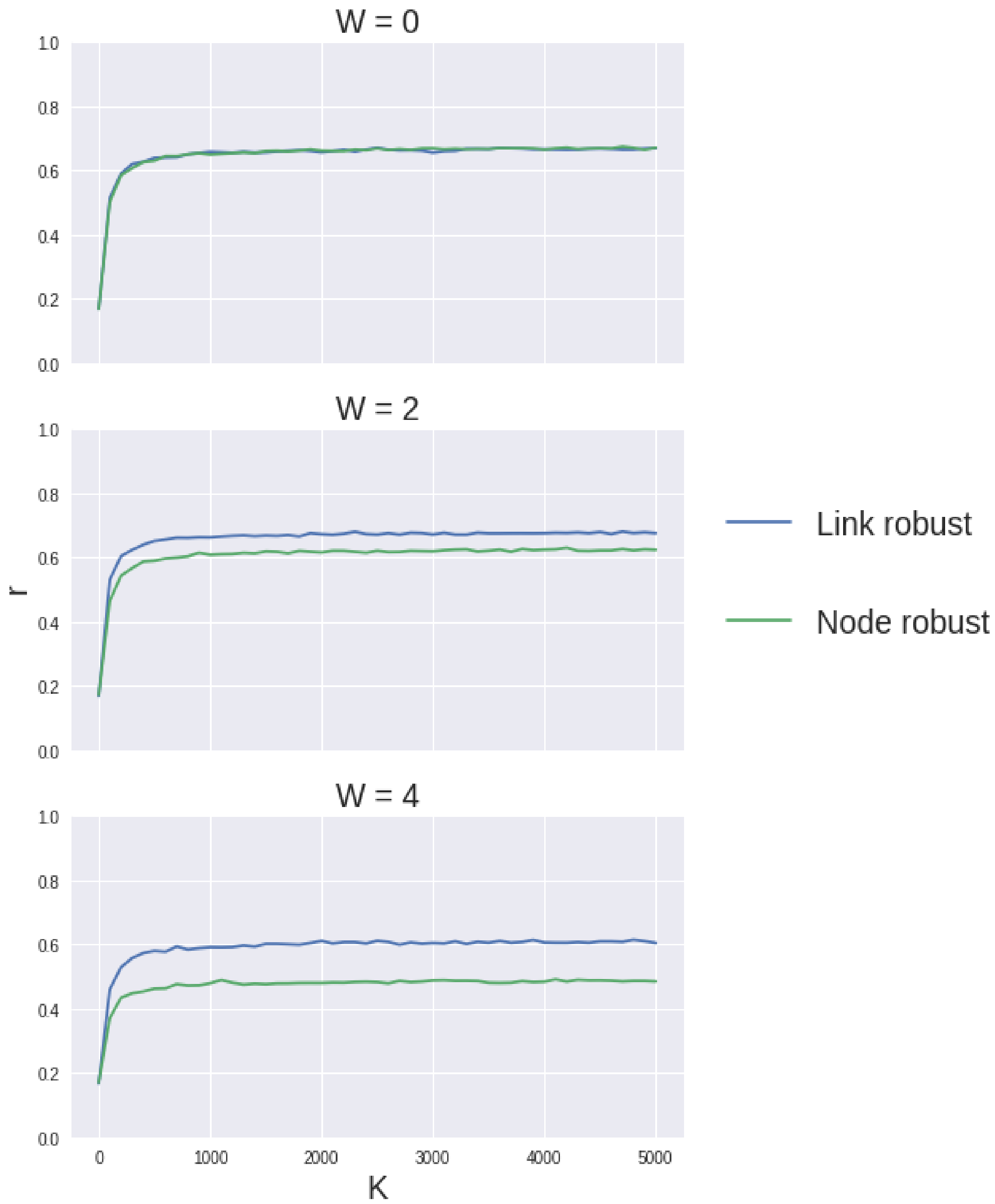

Figure 4.3: The behaviour of order parameter as K increases, for two phases of evolution with different levels of product diversity

middle nodes are the strongly connected components of the network. To prove this, we observe how well these middle nodes synchronize, by first selecting only SCCs from middle node using Tarzan's algorithm, them put them inpo synchrnization scheme. Figure 4.4 illustrates the result. It is obvious that the SCCs synchronize better, compared to the synchronization of the whole network. Another thing we observe from this is that, not all of the evolved networks have SCCs. Table 4.1 shows the number of SCCs founded and the number of original dataset.

Evolution type	W	Number of graphs	Number of SCCs
link robust	W = 0	50	36
	W = 2	40	29
	W = 4	40	32
node robust	W = 0	50	39
	W = 2	40	25
	W = 4	40	6

Table 4.1: Type and number of graphs in the dataset

Another perspective that we want to investigate, is the relation ship between the order parameter and the robustness of evolved network. We measure the order parameter r at $K = 5000$, since r has become stable when K is large, as shown in Figure 4.3. Figure 4.5 shows the relationship between r and rho through different level of product diversity. We also calculate the Pearson correlation for each cases, with the null hypothesis that there is no correlation between ρ and r:

Evolution type	W	Pearson correlation	p-value
link robust	W = 0	-0.801	1.58^{-12}
	W = 2	-0.783	6.22^{-09}
	W = 4	0.134	0.412
node robust	W = 0	-0.229	0.106
	W = 2	-0.167	0.32
	W = 4	0.189	0.265

Table 4.2: Pearson correlation for each type of evolved networks , with different levels of product diversity

We only find the strong negative correlation between the link robust in two cases $W = 0$ and $W = 2$.

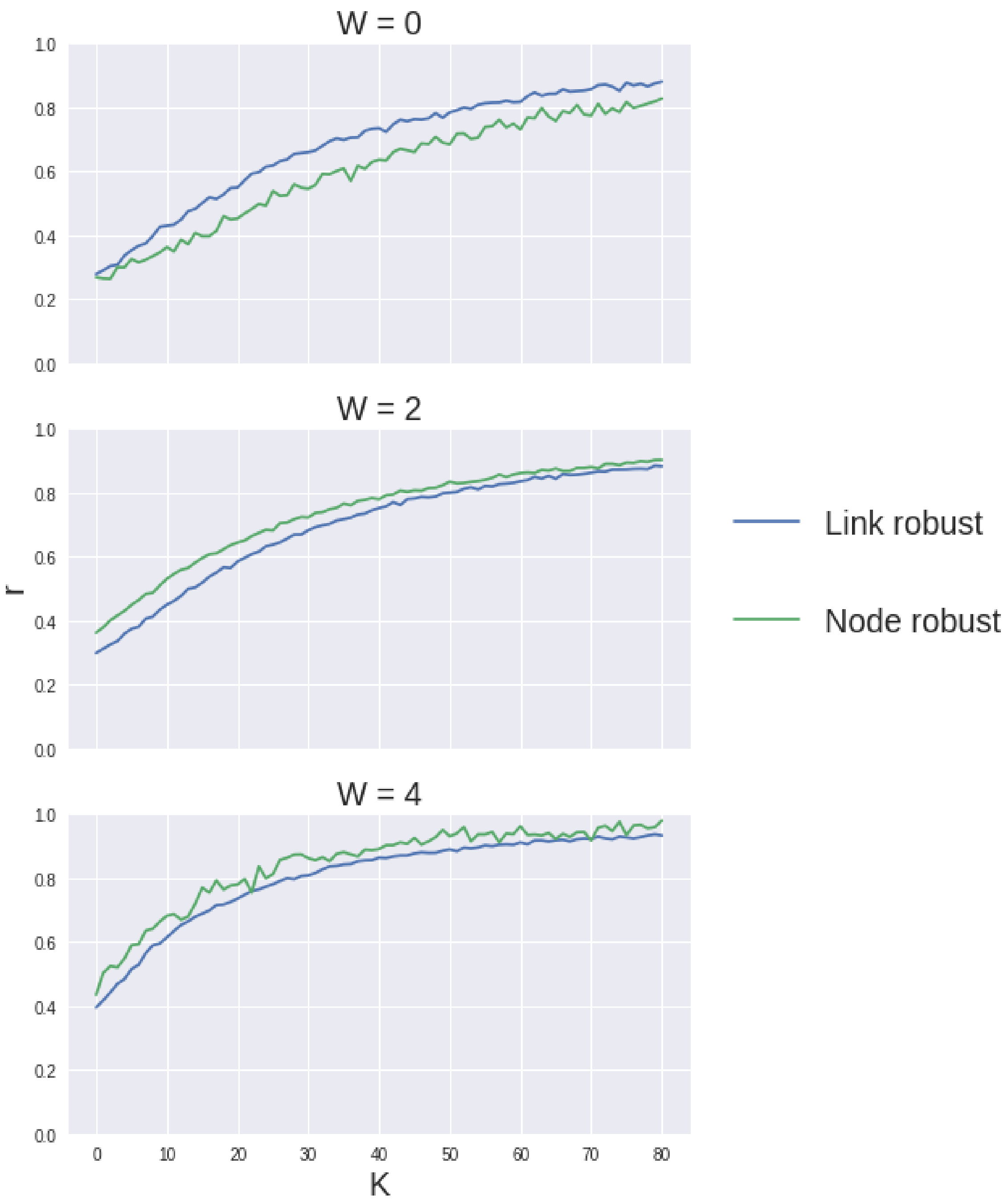

Figure 4.4: The behaviour of order parameter as K increases, for SCCs in two phases of evolution with different levels of product diversity

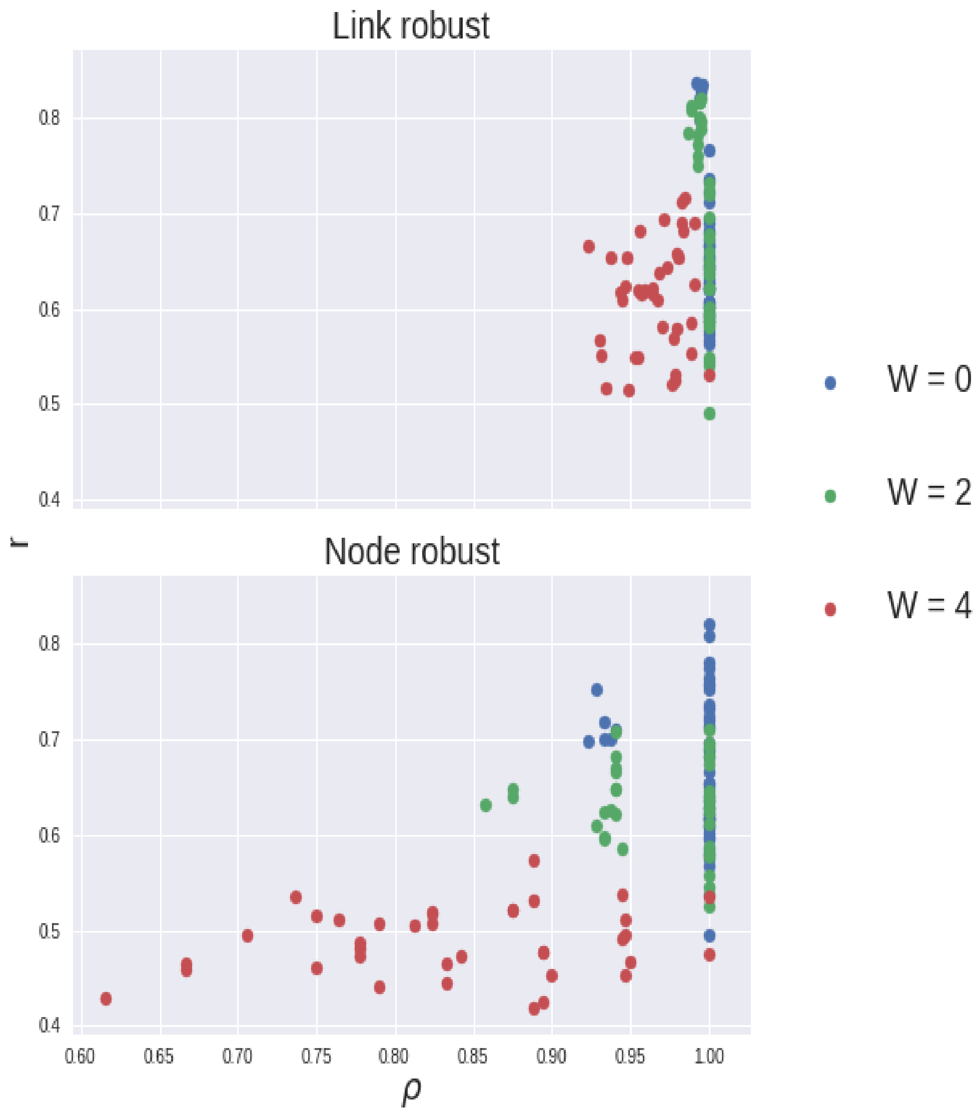

Figure 4.5: The relationship between robustness and order parameter, with different levels of product diversity

5. Conclusions

The purpose of this work is to study how the evolved flow distribution networks synchronize with different levels of diversity. We also study about the structure of these networks in different value of W, as well as the relationship between the order parameter, which is a measure of how a network synchronize, and its number of links.

In Chapter 2 four main backgrounds was introduced: Flow distribution networks, the evolutionary network algorithm to enhance networks' performance as well as robustness to local damages, the Kuramoto model for synchronization and the Tarzan's algorithm for finding SSCs in a given network.

As mentioned in Chapter 3, we create a small database of evolved flow networks using the evolutionary network algorithms introduced in [8, 9, 10] and apply the synchronization scheme. We then go deeper in analyzing the results, by applying the Tarzan's algorithm for finding the SSCs in each network and observing how they synchronize.

The first thing we figure out is the relationship between the number of links in one graph can affect its order parameter r, which measures how well its synchronizes. Applying to the logistic scheme, we can tune the physical synchronization between machines in manufacturing networks by increasing or decreasing connections between them.

We also go to the conclusion that for different values of diversity (represented by the number of zeros W in one column in the prescribed output pattern), the order parameter r is not significantly different. For all cases, networks can not achieve the fully synchronization. The reason for this case is the shortage of interacts between nodes in different layers. It is clear that for input nodes and output nodes, there are no directed links between them. To prove this, we find the SCCs in middle layers of network and apply the synchronization scheme. The result is satisfying: their order parameters r reach to 1.0 very fast, compared to the whole network case. In our small dataset, we also observe that for the node robust case, as W decreases, there are more SCCs that can be found. For the link robust case, the number of SCCs are approximate to each other.

We found negative relationships between robustness and order parameter in link robust scheme, for $W = 0$ and $W = 2$. The rest remains accept the null hypothesis that there is no correlation between ρ and r.

Bibliography

[1] Strogatz, *Exploring complex networks*, Nature 410, 268-276, 2001

[2] R. Albert and A.-L. Barabási, *Statistical mechanics of complex networks*, Reviews of Modern Physics volume 74, 47 (2002)

[3] S. Boccaletti et al., *Complex networks: Structure and dynamics*, Physics Report volume 424, 175 (2006)

[4] E. Salinas and T. J. Sejnowski, *Correlated neuronal activity and the flow of neural information*, Nature Neuroscience. 2, 539 (2001)

[5] P. Fries, *A mechanism for cognitive dynamics: neuronal communication through neuronal coherence*, Trends in Cognitive Science. 9, 474 (2005)

[6] M. Beber, D. Armbruster, M. T. Hütt, *The prescribed output pattern regulates the modular structure of flow networks*, European Physical Journal B, 2013

[7] S. Chankov, M. T. , J. Bendul, *Synchronization in manufacturing system: quantification and relation to logistics performance*, International Journal of Production Research, 2016

[8] P. Kaluza, A.S. Mikhailov, *Evolutionary design of functional networks robust against noise*, Europhysics Letter volume 79, 48001, 2007

[9] P. Kaluza, M. Ipsen, M. Vingron, A. Mikhailov, *Design and statistical properties of robust functional networks: A model study of biological signal transduction*, Physical Review E 75, 15101, 2007

[10] P. Kaluza, M. Vingron, A. Mikhailov, *Self-correcting networks: function, robustness, and motif distributions in biological signal processing*, Chaos 18, 026113, 2008

[11] Y. Kuramoto, *Chemical Oscillations, Waves, and Turbulence*, Springer (1984)

[12] M. Sharir, *A strong connectivity algorithm and its applications to data flow analysis*, Computers and Mathematics with Applications 7(1):67–72, 1981.

[13] R. Tarzan, *Depth-first search and linear graph algorithms*, SIAM Journal on Computing, 1 (2): 146–160, 1972

[14] E. Dijkstra, *A Discipline of Programming*, NJ: Prentice Hall, Ch. 25, 1976

[15] S. Chankov, M. T. Hütt, J. Bendul, *Triggers and effects of synchronization in manufacturing systems*, International Journal of Production Research.