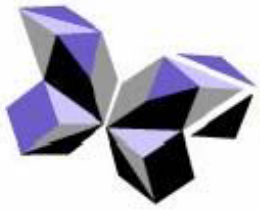

Digital Business Ecosystem

Contract number° 507953

Workpackage 11
Dynamics of Networks

Deliverable 11.2
Report on clustering in networks

Information Society Technologies

Project funded by the European Community under the “Information Society Technology” Programme

Contract number: 507953
Project acronym: DBE
Title: Digital Business Ecosystem

Deliverable Nº: D11.2
Due date: 31/12/2005
Delivery date: 10/02/2006

Short description: A preliminary report on the study of network dynamics and clustering in the second phase of the DBE project.

Partners owning: UBHAM
Partners contributed: STU, LSE, HWU
Made available to: Public

Versioning		
Version	**Date**	**Author, Organisation**
1.0	02/12/2005	Jonathan E. Rowe, Boris Mitavskiy (UBHAM)

Quality check
1st internal reviewer: Philippe de Wilde (HWU)
2nd internal reviewer: Miguel Vidal (SUN)

Contents

Executive Summary

This deliverable is a preliminary report on our work on the dynamics of networks in the second phase of the DBE project. It is divided into three main sections. The first major section describes our continuing work on the problem of estimating the propagation time for a piece of information (for example, a useful service chain) in a network (e.g. of DBE habitats). Some of our earlier work (for example, on the hub and on complete graphs) is briefly repeated, but the main contribution is a general upper bound, which enables us to provide tight estimates for a wide variety of network topologies. This section is largely based on a journal paper currently under submission to Physical Review E.

The second section sets out a proposed framework for looking at hierarchical clustering specifically in network problems. We are focussing on the problem of finding the shortest route between nodes, and the question of when new "short cut"' links should be added to the network in order to gain efficiency of transmission. We will be concentrating on problems of this kind (in collaboration with Miguel Vidal of Sun) in the latter stages of the project.

The third main section concerns general dynamical systems, and reports briefly on the problem of deciding whether or not a particular clustering or "coarse-graining" can be consistently applied. This provides a summary of results obtained in colaboration with Prof. Michael Vose (University of Tennessee) and Prof. Alden Wright (University of Montana) and published in Artificial Life journal [17] and at the Foundations of genetic Algorithms workshop [16]. A complete write-up of these results is under submission to Theoretical Computer Science.

Chapter 1

Introduction and relationship to the project

This deliverable is the first report for sub-task S2 (Dynamics and clustering in hierarchical networks), which is scheduled to run from month 19 to month 36, and builds on the work of S1, already reported [21]. The objectives of this sub-task are:

1. To extend work on information propagation in simple networks to hierarchical ones.

2. To develop theory of structure and dynamics in such networks.

The sub-task interacts with S10 (Symbiosis and competition - HWU) which studies the effects on markets of information sharing in networks, and WP19 (SUN and HWU) looking at the specifics of the DBE peer-to-peer network behaviour (see, for example, [20, 9]). In particular, there is common interest in understanding the effects of different types of network topology, and their effectiveness for certain tasks involving the communication of information. Our research looks at an important abstraction of this question, asking simply how long it might take information to propagate through a network as a function of its size, topology, and the reliability of transmissions. Thus, while our work is mainly theoretical, it is having (and will continue to have) a strong influence on the fundamentals of the DBE structure. In this way, it provides part of the "scientific value chain" linking basic scientific research to the computing domain. Inasmuch as the theory of network dynamics applies to social and business networks, our work also has an influence (albeit indirect) on the understanding of the DBE from a business and social science standpoint.

Chapter 2

Propagation time on stochastic communication networks

2.1 Background

The DBE, at some level of abstraction, comprises a network of nodes or *habitats*. Each of these habitats is an evolutionary environment which provides the associated user with combinations of services to meet their requirements. Services (and their combinations) can also migrate from habitat to habitat — a process in which copies are made (according to their success) and propagate through the network. We seek to understand the dynamics of this propagation, as a function of the network's topology. The situation has an analogue in the spreading of infectious diseases in spatially structured populations.

There are several different questions that can be posed about propagation in networks. If there is only one chance for a node to successfuly transmit to its neighbour, one can ask under what conditions (and with what probability) most or all of the network is "infected" in the steady state. This requires the investigation of the percolation structure of the system [6, 18, 5, 15, 14, 7]. Alternatively, one can focus on the dynamics. A typical approach here is to assume an infinite network with sufficient symmetry to make use of a mean-field approximation in continuous time [10, 4, 13]. In this chapter, we investigate propagation through a *finite* network of arbitrary topology (including so-called *scale-free* networks such as the Internet [22, 19, 3]), assuming that transmissions take place in parallel, in discrete time steps, and with some fixed probability of success. Transmissions are attempted at every time step until successful.

Suppose we have a network of computers and some information is located on one of them. At each time step, processors with the information try to copy it to their neighbours, with success probability p (see figure 2.1). We wish to estimate the expected time for the information to spread to all nodes, which we call the *propagation time*, $E[n]$, on a network containing n nodes. Similarly, one could consider the spread of an infectious disease [2], where p is the infection rate, or the spread of a mutant gene in a metapopulation [11, 8]. The simplest network to consider is a chain of n nodes

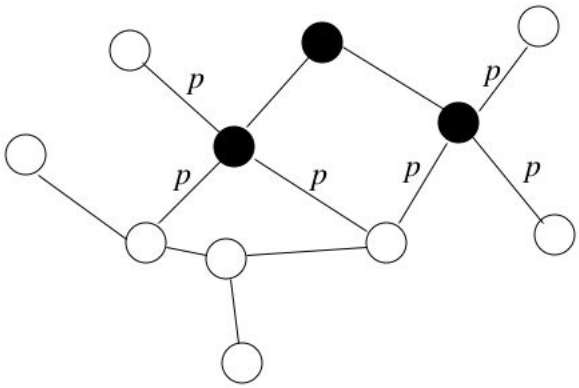

Figure 2.1: Nodes that have the information (black) try, in parallel, to copy to their neighbours. Transmissions succeed with probability p.

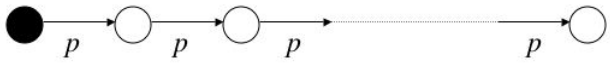

Figure 2.2: A simple chain of n nodes.

(see figure 2.2). If the information starts at one end, then the expected time, $E[n]$, for it to have crossed the network is $(n-1)/p$. This result can be derived from a recurrence relation for the propagation time given k remaining nodes:

$$E[k] = 1 + pE[k-1] + (1-p)E[k] \tag{2.1}$$

A similar recurrence enables us to solve the case of a ring of n nodes. The exact result is complicated but to a good approximation is $(n-1)/2p$.

2.2 The general recurrence and the hub problem

Such a recurrence can be derived for any network: the propagation time starting from a particular situation can be broken down into the possible cases occurring after a single time step, with their associated probabilities.

We consider the general problem of a random sequence $X_1, X_2, \ldots$ of states from some set A, and a subset $D \subseteq A$ of desired states. We can derive a recurrence relation for the first hitting time of the desired subset in terms of all the situations that could possibly arise after a single time step, and their associated probabilities. The first hitting time is defined as $T = \min\{t \mid X_t \in D\}$. In our case, the random sequence comes from the different states of the network as the information (or infection) is propagated from node to node. The desired state is when all the nodes have been infected.

Suppose after the first time step the network is in state $X_1 = k$. We consider the *conditional* probability space that arises from this situation. We write E_k to denote expectation with respect to this conditional probability space, and consider the shifted stochastic process

$$Y_1 = X_2, Y_2 = X_3, \ldots, Y_n = X_{n+1}, \ldots$$

Let $T_k = \min\{t \mid Y_t \in D\}$ be the first hitting time after this first step has been made. We then have:

$$E[T] = \sum_{k \in A} \mathbf{Pr}[X_1 = k] \cdot E_k[T_k] + 1$$

A more complex example is the *hub* in which the information starts at a central node which repeatedly tries to transmit to n neighbours. For example, consider a transmitter signalling to a number of receivers, or a collection of n people that independently have probability p of contracting a disease. The recurrence relation becomes:

$$E[n] = 1 + \sum_{k=0}^{n} \binom{n}{k} p^{n-k} q^k E[k]$$

where $q = 1 - p$. We rearrange to give the recurrence relation:

$$(1 - q^n)E[n] = 1 + \sum_{k=0}^{n-1} \binom{n}{k} p^{n-k} q^k E[k]$$

with $E[0] = 0$. The transmission probability is p and $q = 1-p$. We can use the general recurrrence to prove that $E[n] = \Theta(\log(n+1))$. In particular,

$$\log(n+1) - 1/q \leq \sum_{k=0}^{n} \binom{n}{k} p^{n-k} q^k \log(k+1) \leq \log(n+1) - \log\left(\frac{2}{1+q}\right)$$

(assuming natural logs — for other bases, the constant in the lower bound has to change accordingly).

2.3 Epidemiology and perfect mixing

In epidemiology models of the spread of infectious diseases, it is common to assume *perfect mixing*: that every individual interacts with every other individual. This corresponds to having a complete graph, with every node connected to every other node. The propagation time for a complete graph is bounded by a constant: it does not depend on the number of nodes. Moreover, as n gets large, the propagation time is just two time steps (with increasingly high probability). Suppose we have a complete graph on $n + 1$ nodes and initially one node is infected. The probability that the infection passes to a neighbour is p and we let $q = 1 - p$. After a single time step it is very likely that close to np new nodes have been infected. In fact, Chernoff's inequality [12] tells us that the probability that less than $(1 - \delta)np$ nodes have been infected is

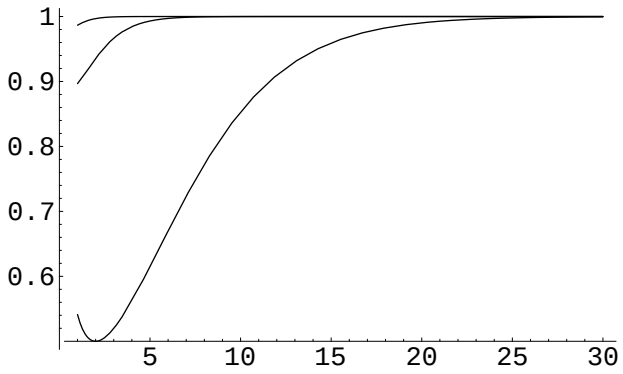

Figure 2.3: The probability that all the nodes of a complete graph are contacted in two time steps, as a function of the number of nodes. The three curves show $p = 0.5, 0.75$ and 0.9.

less than $\exp(-np\delta^2/2)$, where we can make δ as small as we like. This means that, the more nodes there are, the surer we can be that nearly np nodes are infected after one time step. The probability that the remaining nodes get infected on the next time step is therefore close to $(1-q^{np})^{nq}$. Using the fact that $(1-x)^n \geq 1-nx$ for all $0 \leq x \leq 1$, we see that $(1-q^{np})^{nq} \geq 1 - nq^{np+1} \geq 1 - r^n$ (for some r in the range $0 < q^{np} < r < 1$), which is $1 - e^{-O(n)}$.

Similarly for a complete bipartite graph, with n nodes in each set, the probability that a single node in one set infects np nodes in the second gets arbitrarily close to 1 as n gets large. This is then enough to infect all the other nodes in the first set (again with arbitrarily high probability). Then, on the third time step, the remaining nodes of the second step get infected. This analysis can be extended to complete multipartite graphs in an obvious way.

Of course, for actual (finite) systems, one needs to know how big n has to get to obtain this behaviour. This can be modelled as a function of the transmission probability p. Obviously, larger values of p will increase the probability that the transmission will take place within a specific time. For example, figure 2.3 shows the probability that, for a complete graph, the transmission has finished in two time steps, for $p = 0.5, 0.75, 0.9$. It can be seen that, even for a very noisy network ($p = 0.5$), the probability of success within two time steps is near certainty for $n \geq 25$.

In our model, nodes are either infected or not. This corresponds to the SI model of epidemiology (Susceptible-Infected [2]). If I is the number of infected people and $S = n = I$ are the remaining susceptibles, we would like to know how many more people become infected in one time step. For the complete graph (perfect mixing) the number of newly infected people is binomially distributed between 0 and S with success probability $1-(1-p)^I$. If p is small, this is approximately equal to pI and so the expected increase in infected people is close to $\Delta I \approx pSI$, which agrees with the standard SI model. To extend our model to more realistic scenarios, one would have

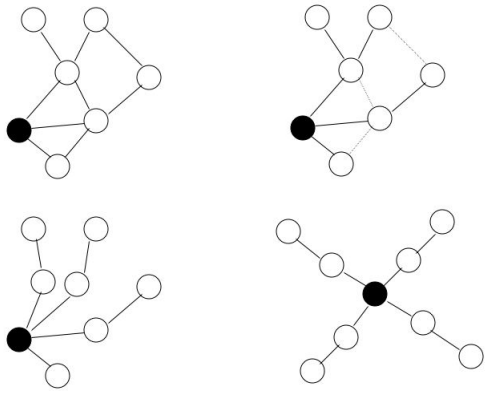

Figure 2.4: To find an upper bound for the propagetion time of a network, we first find a minimum spanning tree. From this, we create a star graph, which we then balance.

to introduce a third state R (removed) for those people who can no longer be infected (due to immunity or death).

2.4 General upper and lower bounds

For the general case, a lower bound on the propagation time, starting from a particular node, is given by the eccentricity of that node (divided by p). That is, the distance from the source to the most distant node in the network. This is because at least that number of successful transmissions will have to be made for the whole network to be infected. It is also possible to derive a general upper bound on the propagation time for an arbitrary network. The idea is to replace the network with a minimum spanning tree, rooted at the starting node. The propagation time on the tree is slower than for the original network, since we may have lost a number of "short-cuts". We then replace the tree with a star graph, with a hub at the starting node, and b branches: one for every leaf of the tree. The length of each branch is the eccentricity, d, of the hub (see figure 2.4). Using Chernoff bounds, we can prove that the propagation time for a star graph is $O((d + \log b)/p)$.

To show this, we return to the general problem of estimating the first hitting time of a desired subset of states in a random sequence. Let A be a set of states and let $D \subseteq A$ be the desired subset. Let $Y(1), Y(2), \ldots$ be a random sequence of states from A that satisfies the following *monotonicity* properties:

1. If $Y(t) \in D$, then for all $k > 0$, $Y(t + k) \in D$.
2. $\mathbf{Pr}[Y(t + k) \in D | Y(t) \notin D] \geq \mathbf{Pr}[Y(k) \in D]$ for all $t, k > 0$.

In other words, the probability of reaching the desired state after a given time interval always improves, and once it is reached, it is never left. In the case of the star graph,

the random variables $Y(t)$ will represent the minimum number of infected nodes along each branch at time t. The desired state is that all nodes on all branches are infected. Clearly, the probability of achieving this state in a fixed amount of time can only improve as time goes by, so the monotonicity condition is satisfied.

Define the first hitting time of the sequence to be $T(D) = \min\{t|Y(t) \in D\}$. Suppose we can find a time τ and constant $0 \leq \varepsilon < 1$ such that

$$\mathbf{Pr}[Y(\tau) \in D] \geq 1 - \varepsilon$$

Then we claim that

$$E[T] \leq \frac{\tau}{1-\varepsilon}$$

That is, if after some time τ (which will in general depend on the structure of the problem), we have some lower bound on the success probability at that time, then we can use this fact to estimate the first hitting time for the whole process.

Proof According to the definition of expectation we have

$$\begin{aligned}
\sum_{k=1}^{\infty} k\,\mathbf{Pr}[T=k] &= \left(\sum_{k=1}^{\tau} k\,\mathbf{Pr}[T=k]\right) + \left(\sum_{k=\tau+1}^{2\tau} k\,\mathbf{Pr}[T=k]\right) + \left(\sum_{k=2\tau+1}^{3\tau} k\,\mathbf{Pr}[T=k]\right) + \cdots \\
&\leq \left(\tau\sum_{k=1}^{\tau} \mathbf{Pr}[T=k]\right) + \left(2\tau\sum_{k=\tau+1}^{2\tau} \mathbf{Pr}[T=k]\right) + \left(3\tau\sum_{k=2\tau+1}^{3\tau} \mathbf{Pr}[T=k]\right) + \cdots \\
&= \tau\left(\sum_{k=1}^{\infty} \mathbf{Pr}[T=k]\right) + \tau\left(\sum_{k=\tau+1}^{\infty} \mathbf{Pr}[T=k]\right) + \tau\left(\sum_{k=2\tau+1}^{\infty} \mathbf{Pr}[T=k]\right) + \cdots \\
&= \tau\,(1 + \mathbf{Pr}[Y(\tau) \notin D] + \mathbf{Pr}[Y(2\tau) \notin D] + \cdots)
\end{aligned}$$

Now the second monotonicity condition can be equivalently stated as:

$$\mathbf{Pr}[Y(t+k) \notin D | Y(t) \notin D] \leq \mathbf{Pr}[Y(k) \notin D]$$

for all $t, k > 0$. Using the definition of conditional probability, this gives us:

$$\mathbf{Pr}[Y(t+k) \notin D] \leq \mathbf{Pr}[Y(t) \notin D]\,\mathbf{Pr}[Y(k) \notin D]$$

We already know that $\mathbf{Pr}[Y(\tau) \notin D] \leq \varepsilon$. And by induction on m we get $\mathbf{Pr}[Y(m\tau) \notin D] \leq \varepsilon^m$. Therefore

$$E[T] \leq \tau \sum_{m=0}^{\infty} \varepsilon^m = \frac{\tau}{1-\varepsilon}$$

as required. □

Now consider propagation in a star graph with b branches of depth d. The problem is equivalent to b parallel repeating Bernoulli trials $X_1(t), X_2(t), \ldots, X_b(t)$, each with success probability p. $X_j(t)$ is the number of infected nodes on branch j at time t.

We want the expected time until all of them have achieved at least d successes. So let $Y(t) = \min_{1 \le i \le b} X_i(t)$, which certainly satisfies the monotonicity requirement. Then

$$\begin{aligned} & \mathbf{Pr}[Y(t) < (1-\delta)tp] \\ = \quad & \mathbf{Pr}[(X_1(t) < (1-\delta)tp) \vee \cdots \vee (X_b(t) < (1-\delta)tp)] \\ \le \quad & \sum_{i=1}^{b} \mathbf{Pr}[X_i(t) < (1-\delta)tp] \\ < \quad & b \exp\left(-tp\frac{\delta^2}{2}\right) \end{aligned}$$

where we have applied Chernoff's inequality. Now we choose time $\tau = \frac{8}{p}(d + \log b)$, and $\delta = 1/2$. Notice that $\tau \ge 2d/p$ and so $d \le \tau p/2$. Therefore

$$\begin{aligned} \mathbf{Pr}[Y(\tau) < d] \quad \le \quad & \mathbf{Pr}[Y(\tau) < \tfrac{1}{2}\tau p] \\ < \quad & b \exp\left(-\frac{\tau p}{8}\right) \\ = \quad & b \exp\left(-d - \log b\right) \\ = \quad & \exp(-d) \\ \le \quad & \exp(-1) \end{aligned}$$

So the probability that we have achieved the desired state by time τ is at least $1 - e^{-1}$. Applying the lemma, we conclude that the expected time to completion is less than

$$\frac{8(d + \log b)}{p(1 - e^{-1})}$$

□

The upper bound for the star is also an upper bound for the original network. Since the eccentricity of any node in a network is less than the diameter D of the network (the length of the greatest distance between nodes of the network), and the number of leaf nodes b is less than n, we have a general upper bound on the propagation time for networks of $O((D + \log n)/p)$. We interpret the diameter D as the time associated with the *depth* of the network, and the factor $\log n$ with the *breadth*. The bound is the maximum of these two factors.

2.5 Results for various networks

We can use these bounds to derive asymptotic results for a range of network topologies. A *random* graph on n nodes is created by assigning an edge between nodes with a given probability. The diameter of such graphs grows as $O(\log n)$. Applying our bound then gives a propagation time of $\Theta((\log n)/p)$. Similarly, it is known that the diameter of various kinds of *small-world* networks also grow at this rate [22], and so also have propagation time $\Theta((\log n)/p)$. In general, *scale-free* networks have diameters that grow at most logarithmically, which means the propagation time may be even quicker for such graphs.

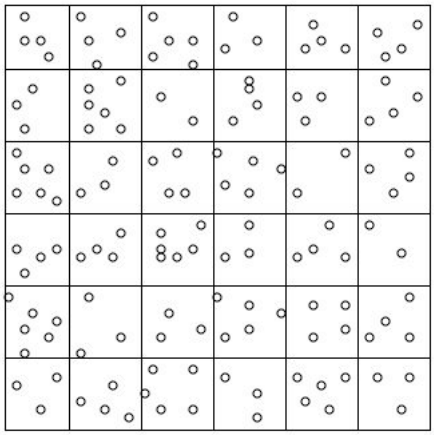

Figure 2.5: Nodes are distributed randomly in the unit square and are connected if they are within a distance of r from each other. We tile the square so that the nodes in each tile form a complete graph.

Hierarchies in organisational structures may be modelled by tree networks. A complete binary tree has depth $\log n$, and so the propagation time, starting at the root, node is again $\Theta(\log n)$. A lattice structure is commonly used in artificial life models (such as cellular automata). Each individual is connected to the neighbours to the north, east, south and west. The diameter of such a network (and therefore the propagation time) is $\Theta(\sqrt{n})$. This is considerably slower than for random and small-world networks. It is known that small-world networks can be constructed from lattices by introducing a small number of random "short-cuts" between nodes [22]. Doing this dramatically reduces the propagation time.

If the diameter grows logarithmically in the number of nodes or faster, then it determines the propagation time (that is, it dominates the "breadth" factor given by the number of branches in the spanning tree). In this case, the propagation time is also inversely proportional to p. However, if it grows slower than logarithmically, we do not get so much information from our bounds. For example, both the complete graph and the hub have constant diameters: our bounds cannot distinguish these cases.

This situation occurs in our final example, in which nodes are spatially distributed. Imagine a square with unit length sides. Nodes are distributed randomly in the square and an edge is drawn between nodes that are less than a distance r apart. For example, this could model a random distribution of processors in a computational lawn that have a limited transmission range [1]. The furthest apart two nodes can be geometrically is $\sqrt{2}$, so the diameter of the network, as n increases, approaches $\sqrt{2}/r$. A lower bound on the propagation time is therefore c/r, for some constant c, which does not depend on n. We will show that the propagation time is also bounded above by a constant and is inversely proportional to r. To do this, we divide the square up into disjoint *tiles* with

side length $r/\sqrt{2}$ (see figure 2.5). The diagonal of each tile is r, so all the nodes in a tile are connected to all the others. The nodes of one tile in isolation form a complete graph, for which the propagation time is constant. Now consider two neighbouring tiles, with a common edge. Place a third tile so that it covers half of each of these. The nodes in the third tile again form a complete graph, with constant propagation time. This means that the expected time for propagation from one of the original tiles to its neighbour is a constant. The situation, therefore, reduces to the constant time spread of information from tile to tile. Since there are $\sqrt{2}/r$ tiles along each side of the square, the number of tiles that have to be traversed on a path between the corners is proportional to $1/r$. Since each move takes place in constant time, the result follows.

In summary, the speed at which information passes through a network is strongly affected by its topology. For a wide range of topologies, the speed is logarithmic in the number of nodes, which is rather fast. In particular, this is the case for randomly connected networks, and certain small world networks. Scale-free networks (e.g. that grow by preferential attachment, such as the Internet, or an unregulated DBE) have propagation time that is at most logarithmic. The time tends to be dominated by the breadth of the network, in these cases, rather than the depth, but even so, communication is rather efficient. Of course, the most efficient times of all can be created by connected every node to every other node, but in practice this is not feasible.

Chapter 3

Clustering in network routing problems

3.1 Introduction

As information, such as service proxies, or messages requesting information propagate through the DBE network, there are many potential routes that it could take. The problem of finding the shortest route between two nodes in a network is, of course, solved directly by the application of Dijkstra's algorithm. However, in the case of the DBE, there are two specific issues that need to be addressed:

1. The topology of the DBE is likely to be *scale-free*, since it will grow by preferential attachments. This will mean that there are a small number of well-connected hubs, through which most traffic will pass. These hubs will then form bottlenecks in the system, slowing down the rate of propagation through the network.

2. The edges that form when a new node joins a network are not necessarily optimal for the messages that will be communicated from and to that node. It may well be that two nodes that are separated by a long path need to communicate frequently. We would like to be able to automatically adapt the network topology to reflect this, and so make communication more efficient.

To study these two problems, we have begun conducting experiments with ways of self-adapting network topologies. We begin by assuming some network topology (e.g. a random scale-free one). Two nodes are then selected, which will try to communicate with each other. Dijkstra's algorithm is used to find the shortest path. We then consider whether or not some kind of *short cut* should be added to the network. In the simple scenarios we have considered, this is done by replacing two contiguous edges by a single edge. To keep the total number of edges constant, we retain just one of the original edges. To simulate the effect that some nodes will communicate much more often than others, we bais the choice of nodes according to some probability distribution. The idea is that, over time, the network will reconfigure itself to enable very short routes between frequently chosen nodes.

To try to prevent hubs becoming bottlenecks, it is necessary to try to avoid them in the routes that are found. We do this by simply making the weight of an edge equal to the sum of the degrees of the nodes that it joins. This means that hubs will have edges coming into them with high weight. They are therefore less likely to be chosen by Dijkstra's algorithm in forming the shortest paths.

The next section describes the initial experiments we have performed, and the results follow. An outline of Dijkstra's algorithm is presented at the end.

3.2 What is the Effect of Rerouting in Scale-Free Networks?

In this section we investigate the following type of problem: Suppose we are given a weighted scale-free network, i.e. we assign a numerical value to every one of the edges of a scale-free network. A percentage of nodes is chosen at random so that the nodes of smaller degree are more likely to be chosen. Now we repeatedly choose a pairs of nodes (the nodes in the pair are chosen independently) and try to find the shortest path between these pairs. The notion of the shortest path can be measured in two ways: it could be *a path consisting of the fewest possible number of edges starting with the first node in the pair* or it could be defined as a *path minimizing the total sum of the weights along it's edges.* Notice that the first definition is a particular case of the second when the weights of every two nodes are the same. Now let's consider the case when the weight between a given pair of nodes is proportional to the sum of the degrees of these nodes. This is the case investigated in the experiment below. Intuitively, this condition makes it more difficult for the shortest path to contain an edge, one, or even more so, both, ends of which happen to be hub nodes. A reasonable heuristic approach to attempt decreasing the average path length is to use the following algorithm: Given a pair of nodes, say (a, b) in the network G, first use Dijkstra's algorithm (please see section 3.3) to find the shortest weighted path connecting a and b. Then once we found the shortest path, starting with the first node on the path, we may decide to combine a consecutive pair of edges with certain probability. If we do so, then one of the intermediate edges having larger weight (if there is a tie than choose the predecessor edge, for instance, or choose uniformly at random) is removed. In other words, if a path $\wp$ contains a fragment $(a,\ b,\ c)$ then with some probability p we may decide to make the following modifications to the graph: add the edge $(a,\ c)$ to the graph (provided this edge is not already in the graph). Remove either the edge $(b,\ c)$ or the edge $(a,\ b)$ depending on which one of these two has a bigger weight. If they both have the same weight, then remove either one of them at random. We investigate several aspects of this procedure depending on the probability p of "concatenation" of edges and the percentage of nodes from which the starting and the ending nodes are sampled.

1. What is the relationship between the number of edges in the shortest paths before and after the experiment?

2. What is the relationship between the total weights in the shortest paths before and after the experiment (upon the completion of the experiment we reset the weights to be proportional to the sum of the degrees of their ends)?

Moreover, we introduce to measures of complexity of Dijkstra's computation, which are the total number of nodes encountered when carrying out the computation and also the total number of nodes to which the shortest path has been found during the computation of the shortest path between a and b. We repeat the simulation (sampling of the random nodes a and b from the network and finding the shortest path between them 50 times and compute the arithmetic means of all the quantities mentioned above. Then we start the iteration procedure of replacing a pair of consecutive edges in the shortest path found by the Dijkstra's computation for the pair $(a,\, b)$ with their concatenation. We carry out this procedure 300 times (due to limited computer power when the network involves a large number of edges and nodes). Then, again we repeat the sampling of the pairs of nodes from the initially chosen set in the modified network (consisting of the same collection of nodes but different edges) 100 times. We compute the same exactly quantities as for the initial experiment. The figures in the next section show the ratio of the corresponding quantities before and after the experiment.

The parameters chosen for the experiment were as follows: p was chosen to be either either 0.2 or 0.8 and the percentage of nodes chosen was either 5% or 10%. There were 4 independent experiments conducted (1 experiment for every pair of the probability and percentage values mentioned above). The outcome of the experiment (as the reader can see from the figures in the next section) suggests that the experiment has any observable effect on reducing either of the quantities considered (the average number of edges in the shortest path between the chosen pairs, the average weight of the shortest paths and either one of the chosen computational cost measures) only when the size of the set of nodes from which we sample is small (i.e. when $\frac{pr\cdot\text{number of nodes in the network}}{100}$ is a small number where pr denotes the percentage of nodes selected). In fact, interestingly enough, from the data one can see that the other parameters don't play a particularly significant role since the ratios of all the quantities tend to 1 as the set of nodes between we look for the shortest path increases beyond a certain threshold. We are still working on finding an explanation for this phenomenon but it is probably due to the fact that after concatenating edges and removing others at the same time the network retains its scale-free type of structure. On the other hand, when the number of nodes between we are looking for the shortest path is small, the procedure concatenates relatively few paths so that the set of paths which are shortened didn't share enough edges in the original network. It is worth mentioning that increasing the probability of concatenating consecutive edges in a shortest path between a requested pair of nodes makes the algorithm effective for a larger subset of possible requested nodes. We are currently attempting to discover a rigorous foundation for these conclusions. Moreover we plan to continue running similar experiments with the aim of discovering other, perhaps more effective algorithms depending on their purpose.

3.3 Results

The first set of results concerns the (unweighted) lengths of the paths found between communicating nodes. As new edges get added to the network (replacing old ones), then frequently communicating nodes should come closer together. Figure 3.1 depicts this process for networks of increasing size. Figure 3.2 presents similar results, this

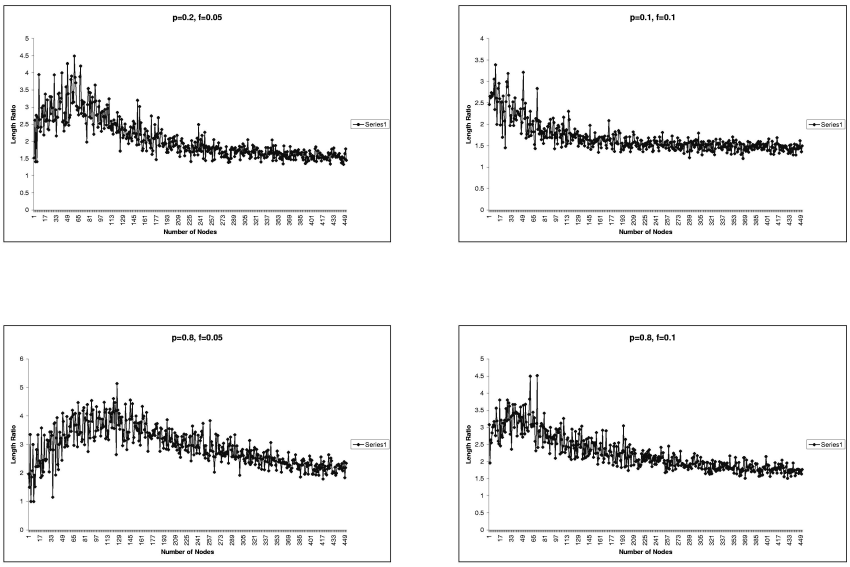

Figure 3.1: The ratio of shortest path lengths found before applying the short-cut algorithm to lengths found after running the algorithm for 300 iterations, for different combinations of p (probability of adding short cut) and f (fraction of nodes being selected).

time giving the ratio of improvement of the weights of the shortest paths found. It can be seen that considerable improvement can be made for relatively small networks, if a large probability of short-cut insertion is used. As the network size grows, however, the savings become less (improvement ratio tends to one). This could simply be because the algorithm has not run for enough iterations, or because there are now more pairs of nodes being selected from, so improvements to some routes create problems for others. We seek to investigate this further.

We also measure the increase in efficiency of running Dijkstra's algorithm, which should improve as commonly chosen nodes are brought closer together in the network. Figure 3.3 shows the total number of nodes the algorithm expands in its search. We plot the ratio of before the short-cut algorithm is applied to after 300 iterations. We again see considerable improvements in efficiency for relatively small networks, although this gain decreases as the network size grows.

Appendix: Description of Dijkstra's Algorithm

Dijkstra's algorithm is a procedure used for finding the shortest path between a given pair of nodes. The algorithm works as follows:

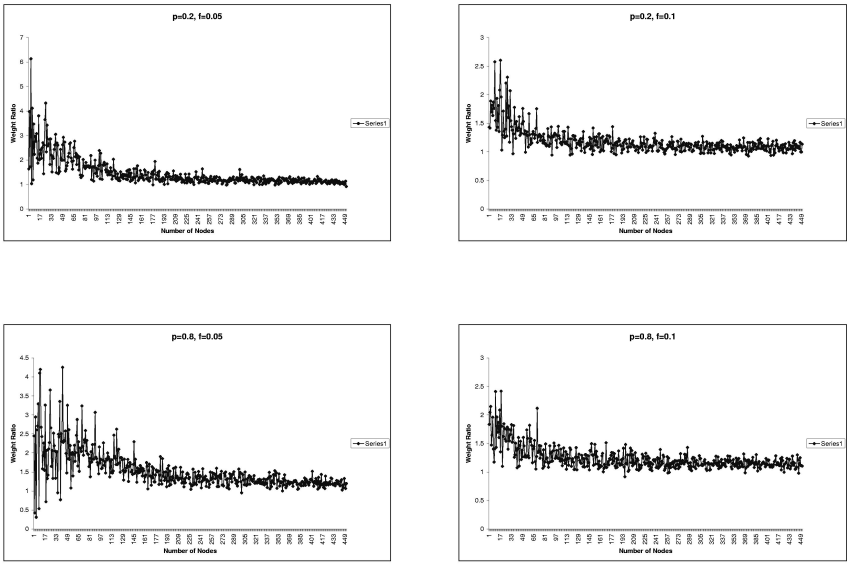

Figure 3.2: The ratio of the weights of the shortest paths found before applying the short-cut algorithm to weights found after running the algorithm for 300 iterations, for different combinations of p (probability of adding short cut) and f (fraction of nodes being selected).

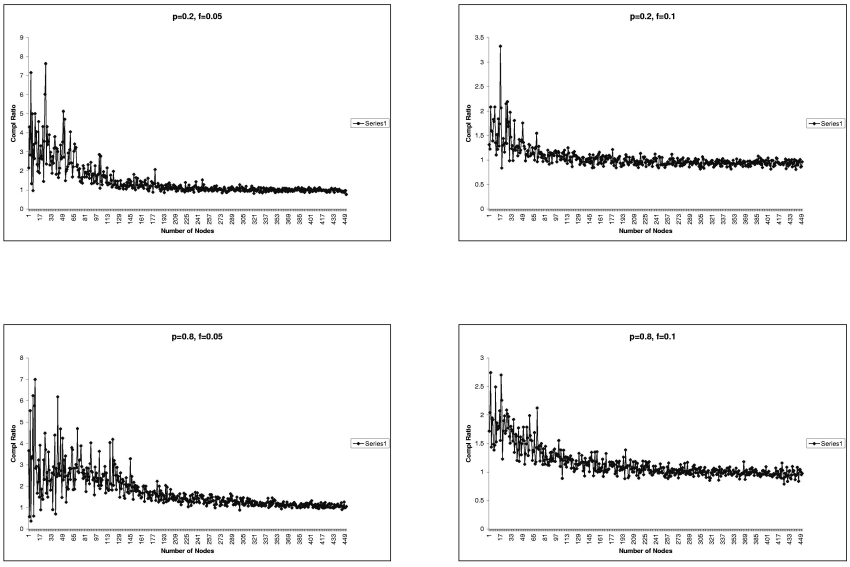

Figure 3.3: The ratio of the efficiency of Dijkstra's algorithm before applying the short-cut algorithm to that after running the algorithm for 300 iterations, for different combinations of p (probability of adding short cut) and f (fraction of nodes being selected).

Input: A pair of nodes $(a,\ b)$.
Output: A shortest weighted path from a to b.
Procedure: At every step, the algorithm keeps record of the following information:

1. An array of nodes.

2. For every node in the array, it keeps track of the current estimate of the minimal path length from a to that node estimated up to date. If the value of that path weight (call it $w(x)$ for the given node x) is known to be minimal (we shall tell below what the criteria for this is) then the node is declared to be "tight" (i.e. the corresponding boolean variable $tight = true$. Otherwise $tight = false$).

The algorithm starts by initializing the array having currently only one node a which is declared to be tight. The weight from a to a is 0.

At every step the algorithm looks for all possible neighbors of the node which has been declared tight in the previous step (let's call it x_t) and adds them to the array if they are not in the array already). All the newly added nodes are NOT tight. The weight of every newly added node, call it y, is set to $w(x) + w(x,\ y)$ and the weights of the neighbors of x_t which have been present in the array are updated to $w(x) + w(x,\ y)$ provided their current weight is less than $w(x) + w(x,\ y)$ where $w(x,\ y)$ denotes the weight of the edge from x to y. The algorithm then declares exactly one of all the nodes currently present in the array tight. This is a node having the least possible estimate for the length to it from a among the nodes which are not tight at the present time. It can be proved that this value is, indeed, the desired minimal length. The algorithm is repeated until the node b is added to the array *and* declared to be tight.

Chapter 4

Hierarchical dynamics

4.1 Brief description

Suppose we have a system whose state can be described by n variables. That is, it corresponds to a point in n-dimensional phase space. We also have some map $h : \mathbb{R}^n \to \mathbb{R}^n$ which describes how the system changes through time. That is, if $x(t) \in \mathbb{R}^n$ is the state at time t then $x(t+1) = h(x(t))$. In many such systems, one can sometimes find ways to reduce the number of degrees of freedom of the system. That, one might have a map $\Xi : \mathbb{R}^n \to \mathbb{R}^m$ where $m < n$, so that $\Xi(x)$ represents a "coarse-graining" of x. Under some mild assumptions (e.g. that h and Ξ are smoothly differentiable) we investigated the question of determining when a map h is *compatible* with a coarse-graining Ξ. Intuitively, this means that one can calculate the trajectory of points in the reduced space $\mathbb{R}^m$, without reference to the original phase space. Formally, the question is whether or not there exists a map $\bar{h} : \mathbb{R}^m \to \mathbb{R}^m$ such that $\bar{h}(\Xi(x)) = \Xi(h(x))$ for all x.

The situation is simplest when Ξ is a linear operator. In this situation, the coarse-graining is simply a linear combination of variables. This situation is explored in [17] where some example applications are also given. The necessary and sufficient condition for a compatible coarse-graining is that the kernel of the operator Ξ (that is, the set of points which $'Xi$ maps to the origin) is an invariant subspace of the differential of h (evaluated at any point in the phase space). The details are worked out in the cited paper, but in summary, this gives us a powerful way of determining what linear coarse-grainings are possible for a dynamical system, simply by solving a differential equation, or in the case of linear dynamics, by applying a theorem concerning the group of symmetries acting on the dynamics.

The case of non-linear projections, Ξ, is somewhat more complicated, and is treated in [16]. The first complication is that for points in the phase space to be considered equivalent, they not only have to mapped to equal points under Ξ, but they have to belong to the same connected level set. This condition is automatically met for the linear case. The second complication is that it is now the kernel of the differential of Ξ that plays the role of the invariant subspace.

4.2 Example

As an illustration of the group theory technique, consider the movement of a service proxy from node to node in the DBE. Unlike in the previous sections, we now consider the case where the service moves, rather than being copied. Suppose that edges between nodes have a weight which represents the strength of association between the corresponding habitats. When the service is choosing which way to migrate, it picks a neighbouring node with probability proportional to the associated edge weight. We might wish to analyse the dynamics of the service as it moves around the network, and also consider which nodes (or habitats) might profitably be clustered together.

We describe the progress of the service as a Markov chain, in which the habitats of the DBE correspond to the different states the service can be in. According to the probabilistic rule of choice, the probability that the service moves from habitat j to habitat$i$ is

$$\frac{w_{i,j}}{\sum_k w_{k,j}}$$

where $w_{i,j}$ is the weight of the edge from j to i. These probabilities determine the transition matrix Q of the Markov chain. Now consider a map π which is a permutation of the habitats, which has the property of preserving edge weights. That is,

$$w_{i,j} = w_{\pi(i),\pi(j)}$$

for all i and j. The map is called an *automorphism* of the weighted network, and the set of all such automorphisms forms a group that acts on the nodes of the network. We define a corresponding permutation matrix σ_π by

$$(\sigma_\pi)_{i,j} = \begin{cases} 1 & \text{if } i = \pi(j) \\ 0 & \text{otherwise} \end{cases}$$

We now show that this permutation matrix commutes with the transition matrix Q.

$$\begin{aligned} (\sigma_\pi Q)_{i,j} &= \sum_k (\sigma_\pi)_{i,k} Q_{k,j} \\ &= \sum_k [i = \pi(k)] \frac{w_{k,j}}{\sum_m w_{m,j}} \\ &= \frac{w_{\pi^{-1}(i),j}}{\sum_m w_{m,j}} \\ &= \frac{w_{i,\pi(j)}}{\sum_m w_{m,j}} \end{aligned}$$

and

$$\begin{aligned}
(Q\sigma_\pi)_{i,j} &= \sum_k Q_{i,k}\,(\sigma_\pi)_{k,j} \\
&= \sum_k \frac{w_{i,k}}{\sum_m w_{m,k}}[k = \pi(j)] \\
&= \frac{w_{i,\pi(j)}}{\sum_m w_{m,\pi(j)}} \\
&= \frac{w_{i,\pi(j)}}{\sum_m w_{\pi^{-1}(m),j}} \\
&= \frac{w_{i,\pi(j)}}{\sum_m w_{m,j}} \\
&= (\sigma_\pi Q)_{i,j}
\end{aligned}$$

The Group Orbit theorem now says that if we take any subgroup of automorphisms (that is, any collection of automorphisms that is closed under composition, taking inverses, and that includes the identity map), then the orbits of such a subgroup can be used as a way of clustering the nodes so as to respect the dynamics of the system. This means that two nodes i and j can be clustered together if there is an automorphism such as π in our chosen subgroup such that $i = \pi(j)$.

4.3 Summary

The theorems we have proved provide very general characterisations of the conditions under which a particular clustering (or coarse-graining) will respect the underlying dynamics of the system. We have begun to consider ways in which these ideas might be applied more directly to the problems raised by the DBE. In particular, our experimental results concerning adding short-cuts to Dijkstra's algorithm indicates that some more sophisticated method is needed to decide when to add (or remove) edges to the network, to help it self-adapt. It remains an open question whether or not the mathematical characterisation of legal coarse-grainings will be able to assist in this matter.

Bibliography

[1] H. Abelson, D. Allen, D. Coore, C. Hanson, G. Homsy, T. F. Knight, R. Nagpal, E. Rauch, G. J. Sussman, and R. Weiss. Amorphous computing. *Communications of the ACM*, 43:74–82, 2000.

[2] R. M. Anderson and R. M. May. *Infectious diseases of humans: dynamics and control*. Oxford University Press, 1991.

[3] A.-L. Barabási. The physics of the web. *Physics World*, 14:33, 2001.

[4] M. Barthélemy, A. Barrat, R. Pastor-Satorras, and A. Vespignani. Velocity and hierarchical spread of epidemic outbreaks in scale-free networks. *Physical Review Letters*, 92:178701, 2004.

[5] I. Benjamini, G. Kalai, and O. Schramm. First passage percolation has sublinear distance variance. *Annals of Probability*, 31:197–208, 2003.

[6] P. Grassberger. On the critical behavior of the general epidemic process and dynamical percolation. *Mathematical Biosciences*, 63:157–172, 1983.

[7] G. Grimmett. *Percolation*. Springer-Verlag, 1989.

[8] I. Hanski and O. E. Gaggiotti. *Ecology, genetics, and evolution of metapopulations*. Elsevier Academic Press, 2004.

[9] HWU. Derivation of network topology from combining graph-theoretic and system requirements. DBE Deliverable 19.2.

[10] M. Keeling. The implications of network structure for epidemic dynamics. *Theoretical Population Biology*, 67:1–8, 2005.

[11] E. Lieberman, C. Hauert, and M. A. Nowak. Evolutionary dynamics on graphs. *Nature*, 433:312–316, 2005.

[12] M. Mitzenmacher and E. Upfal. *Probability and computing*. Cambridge University Press, 2005.

[13] M. E. J. Newman. Spread of epidemic diseases on networks. *Physical Review E*, 66:0161128, 2002.

[14] R. Pemantle and Y. Peres. Critical random walk in random environment on trees. In G. Grimmett, editor, *Probability and Phase Transition*, pages 261–264. Kluwer Academic Publishers, 1994.

[15] R. Pemantle and Y. Peres. Critical random walk in random environment on trees. *Annals of Probability*, 23:105–140, 1995.

[16] J. E. Rowe, M. D. Vose, and A. H. Wright. Coarse-graining selection and mutation. In A. H. Wright, M. D. Vose, K. A. De Jong, and L. M. Schmitt, editors, *Foundations of genetic algorithms (FOGA-8)*, pages 176–191. Springer, 2005.

[17] J. E. Rowe, M. D. Vose, and A. H. Wright. State aggregation and population dynamics in linear systems. *Artificial Life*, pages 473–492, 2005.

[18] L. M. Sander, C. P. Warren, I. M. Sokolov, C. Simon, and J. Koopman. Percolation on heterogeneous networks as a model for epidemics. *Mathematical Biosciences*, 180:293–305, 2002.

[19] C. Song, S. Havlin, and H. A. Makse. Self-similarity of complex networks. *Nature*, 433:392–395, 2005.

[20] SUN. Dbe nervous system. DBE Deliverable 19.1.

[21] UBHAM. Report on the flow of software components in a static network. DBE Deliverable 11.1.

[22] D. J. Watts. *Small worlds: the dynamics of networks between order and randomness*. Princeton University Press, 2004.