Escola Politècnica Superior de Castelldefels

UNIVERSITAT POLITÈCNICA DE CATALUNYA

BSc THESIS

TITLE: Modelling of traffic matrices with multiresolution analysis techniques

TITULATION: Enginyeria Tècnica de Telecomunicació, especialitat Telemàtica / BSc in Telecommunications, Computer Networks

AUTHOR: Josep Xavier Torres Haba

DIRECTOR: David Rincón Rivera

DATE: October, 16th 2009

Títol: Modelització de matrius de trànsit amb tècniques d'anàlisi multiresolució

Autor: Josep Xavier Torres Haba

Director: David Rincón Rivera

Data: 9 d'octubre de 2009

Resum

Les matrius de trànsit consisteixen en un conjunt de dades que permeten caracteritzar una xarxa relacionant cada parella de node ingrés i node de sortida amb el volum de trànsit que hi circula. Entre d'altres coses, les matrius de trànsit ens permeten modelar la demanda de trànsit, detectar anomalies, predir el comportament d'una xarxa i configurar-la adientment.

En aquest treball de fi de carrera s'analitzen mostres d'aquestes matrius de trànsit, veient algunes de les seves aplicacions i característiques (detecció d'anomalies, balanceig de càrrega, la seva evolució en el temps, etc), a més de presentar la manera com podem obtenir aquestes dades, quins són els problemes més típics en aquesta presa de mesures i com actuar quan les dades son incompletes (inferència), tot i que aquest tema no sigui l'objectiu principal.

Amb l'objectiu de trobar nous, i bons, models de matrius de tràfic utilitzem les denominades *wavelets de difusió*, una variant relativament nova de les transformades wavelets tradicionals que ens permeten caracteritzar una xarxa i veure la "difusió" dels fluxos de tràfic sobre la mateixa. Per això hem de construir un operador de difusió fiable que ens permeti obtenir millors resultats que els que obtenim amb els models usats actualment. Presentarem alguns operadors existents i proposarem un nou operador que també té en compte la correlació temporal que hi ha en les sèries de mostres. Un cop obtingut l'operador veurem com afecta a la difusió sobre alguns exemples i analitzarem la compressibilitat i l'error subseqüent que indueixen.

Els resultats dels experiments s'han obtingut amb dades d'Abilene (xarxa nord-americana) i de GÈANT (xarxa acadèmica europea), xarxes que degut a la seva magnitud treballen amb una gran quantitat de dades i sobre les quals resulta molt més interessant trobar patrons de comportament que permetin predir-los.

Title: Modelling of traffic matrices with multiresolution analysis techniques

Author: Josep Xavier Torres Haba

Director: David Rincón Rivera

Date: October, 9th 2009

Overview

Traffic Matrices are datasets which allow us to characterize a network by relating each pair of ingress-egress nodes with the volume of traffic flowing over them. Among others, traffic matrices allow us to detecting anomalies, predicting its future evolution and configuring it properly.

In this thesis we analyze some traffic matrices and we see some of their features and applications (anomaly detection, routing and load balancing, their time evolution, etc), as we present how to get these measures, which the most common problems we will have to face at the time of getting these measurements are and how to infer data from real measurements when they are incomplete, although these are not our main objectives.

With the objective of finding new, and reliable, traffic models, we used *diffusion wavelets*, a new variant of traditional wavelet transforms, which allows us to characterize a network and see the diffusion of the traffic flows over it. For this purpose, we have to build a reliable diffusion operator which allows us to get better results than the ones we obtain with the models used today. We will present some existing operators and will introduce in our new operator the idea of the time correlation between the samples series. Once we get the operator, we will see how the diffusion is developed over some examples and we will analyze the compressibility and its subsequent induced error afterwards.

The results of the experiments have been obtained from Abilene data (north-American network) and GÉANT's (European academic network), networks that, because of their magnitude, work with a lot of data. This makes more interesting and easier to find behaviour patterns which allow some sort of predictions. Moreover, the fact that there are whole networks hanging from these supernodes, gives us the chance to get some conclusions about our approach.

INDEX

INTRODUCTION

In this thesis we will introduce the use of traffic matrices (TMs) in communications engineering. The objective is to analyse them in order to find patterns that allow us to build a predictive model. Besides, we will introduce the use of diffusion wavelets and the associated operators for this purpose, and will study the compressibility and errors resulting from the application of this technique.

In the first chapter the reader can find information about traffic matrices, which are a dataset that measures of the traffic flowing over a network. As they will be our basic tool, we must review the procedures to obtain them and to work with them. We will present two ways to get these matrices: Netflow and SNMP. We will also see that the main problem brought by traffic matrices is not related to engineering: public data is almost unavailable. So, even though it is not our main objective, we will have somehow to infer some data for working. Additionally, we will see the TM properties and visual representations.

The second chapter will present the theory of wavelets and the basics of how we will apply them in order to perform a multiresolution analysis (MRA). Here we will find the mathematical description of wavelets. These maths will lead us to the multiresolution analysis and we will see how it is done. The reader can also find the description of a new kind of promising wavelets: diffusion wavelets, which we will use to perform our MRA over a graph.

In the third and last chapter a detailed study of the application of Diffusion Wavelets to traffic matrices can be found. First, we will focus on the importance of diffusion operators and how to build a new one. We will describe the first experiments done with simple operators. Later, we will add temporal correlation to our operator, expecting to take advantage of the inherent correlation in the traffic series. Our hypothesis is that the temporal evolution of the flows over a network may follow some pattern. Furthermore, this behaviour may let us predict the fluctuations of the flows in the future. After the diffusion process, we have also computed the reconstructions to discover their compressibility and their error, because it is important to be able to undo the process to see if we can recover original data.

Finally, we can see the conclusions we got from the results. We found that there is some sort of time correlation, but the results have shown it may not be the link to get a better model. However, this research may be the first step towards an improvement of the existing models. In addition, we will also list some future lines of research that would lead us to a more complete knowledge of this topic.

To summarize, the main contributions of this thesis have been firstly, the confirmation of the results of previous works, secondly the definition of the 3D operator with time correlations, and thirdly the analysis of the results obtained with this 3D operator.

The annexes contain extra information (code, extended results, graphs and the paper [10] presented in JITEL 2009, Cartagena, Spain, with some results of previous work in this topic) and are referenced along the document, as well as the papers, books or webs in the bibliographic references.

CHAPTER 1. TRAFFIC MATRICES

1.1. Definition

Traffic matrices (TMs) are a way to describe the volumes of traffic exchanged by the ingress and egress nodes in a network. These nodes can be either simple nodes or Points of Presence (PoPs[1]), from where there hang other nodes or even whole networks. For each pair of nodes (in-out) the TM specifies the amount of traffic flowing over a path between the nodes during a certain time slot. In our datasets, this interval is, for Abilene[2] (the north-American academic network) five minutes and for GÉANT[3] (the European academic network) fifteen minutes. Note that the traffic matrix does not have to be mandatorily symmetric. Additionally, one point of interest will be the generated traffic by a node towards itself: that is why we have values for the X-X pairs too (for example, Seattle-Seattle or LA-LA). This is acceptable if the node is a PoP, since part of the traffic generated by customers attached to the PoP may go to other customer also attached to the same PoP.

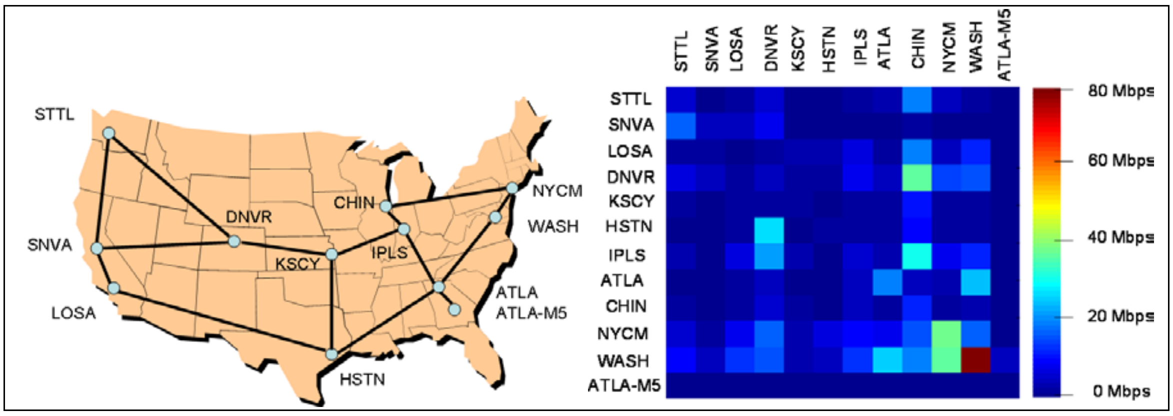


Fig. 1.1 Representation of Abilene (USA) and graphic representation of one of its traffic matrices (2004 March, 3rd from 12:00 to 12:05).

We can see in figures 1.1 and 1.2 the result of displaying a traffic matrix as if it was an image, with the origin in the y axis and the destination in the x axis.

[1] PoP: A Point of Presence is an artificial interface between communication entities,that provides, in our case, access to the Internet. It houses servers, routers, aggregators, etc. It uses to be part of the facilities of a telecommunications provider, rented by an ISP (Internet Service provider). An ISP usally is compsed by several Points of Presence. [31]
[2] http://en.wikipedia.org/wiki/Abilene_Network
[3] http://en.wikipedia.org/wiki/G%C3%89ANT, http://www.geant.net/

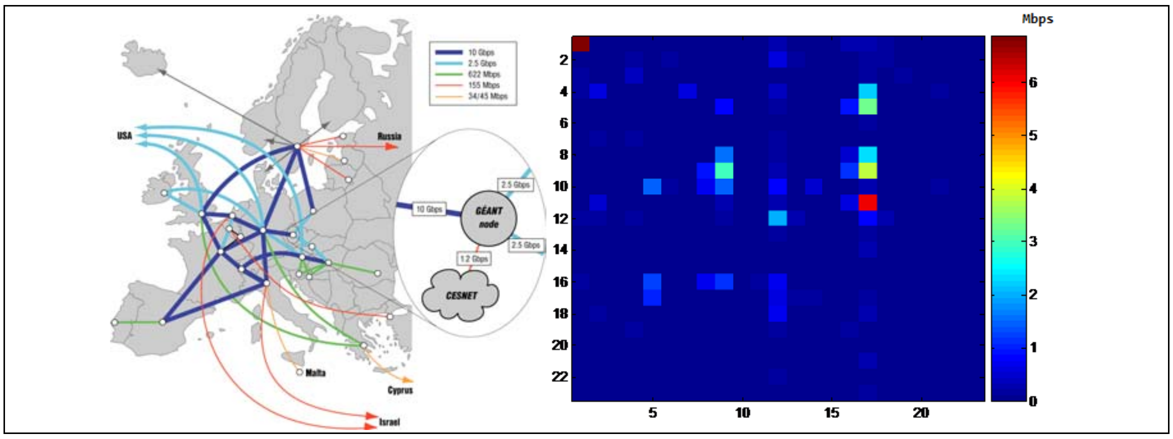


Fig. 1.2 Representation of GÈANT (Europe) and graphic representation of one of its anonymized traffic matrices (2005 February, 22nd from 19:45 to 20:00).

1.2. Importance of Traffic Matrices

Traffic matrices have a great potential utility for IP network management and capacity planning since they relate three basic parameters of a network: origin, destination and volume of traffic. This kind of matrices is used in traffic engineering since they try to get optimum performance from the routing in terms of, for example: load balancing, quality of service, etc.

Given the importance of TMs, the availability of good models is essential for the operators of networks because they will give them good matrix estimations and the capability to apply these estimations daily in their operations.

With the information they provide we can design a new network, detect anomalies, simulate data and predict the network behaviour in a specific time. That is why there is a lot of interest in the research community about modelling traffic matrices and predicting them.

1.2.1. Applications of Traffic Matrices

The fact is that operators have always looked (and they still do) for getting better estimating models or new and more accurate solutions for the problems that a traffic flowing over a link can produce: planning, failures, gathering information for monitoring, detecting *a priori* unpredictable fluctuations... So we can state that there are four main goals behind the TM analysis:

- **Synthesis:** One of the first troubles we find is that we need huge quantities of samples (TMs) for determining a reliable simulation of, for example, a new network or a new protocol. Once we achieve a synthesis of TMs, we can create traffic matrices from which we can also design and build entire networks (localisation of routers or PoPs, links and assignment of capacity). The synthesis must be as reliable as possible (in order to minimize the error between the synthesized and the original samples) [17, 18].

- **Routing and load balancing** [1]**:** Another important application of TMs is to optimize the routing. In traffic engineering we look for the optimization of the routing performance in terms of load balancing, quality of service (QoS), etc. There may be links with high aggregated loads that slow down the traffic: it can be solved by setting up alternative routes to balance the loads over the nodes in our network. For this purpose, there could be great if we had some model to predict the variations of demand in the long and medium term. If we had this information the operator would be able to reconfigure the net to absorb these peaks.

- **Detection of anomalies** [11]**:** There are different anomalies a network operator can be interested in detecting. On the one hand there is the probability that a group of users (or just one) show a change of behaviour, for example, increasing their traffic demands. In this case, our model should be able to point out that there is something happening (so that the network could answer). On the other hand, a hostile user or some mismatch on the network may cause a lot of petitions to a specific node. This is called a Deny of Service attack (DoS) and (if provoked) it is used to block some service or host by overloading it with incoming traffic. In this other case, the model will show us that something is being drawn away from the usual pattern. Regarding to load balancing, detecting these anomalies can also help to reconfigure the network routes and loads.

- **Prediction models:** One of the most important research topics is to find a model that provides a reliable prediction of the future TMs. We have already commented the importance of this model for ISPs. The fact of getting ahead the changes, and acting before they pass would be a great achievement. Unfortunately, there is no much general knowledge on how to get to a reliable model, so most research working is being done in these topics [1, 6].

1.3. Obtaining Traffic Matrices

We said that traffic matrices are a powerful measurement for traffic engineering and network planning. However, troubles appear in one of the first steps, since it is not trivial to get these measurements.

The first problem is how to measure real data in real time. For this purpose we can use two procedures:

- NetFlow [28] is a network application developed by Cisco (there are similar products from other fabricants). Among all of its features, this protocol can measure flows and traffic volumes passing through a router. The main problem is that when a node receives lots of flows, it requires a lot of CPU power to process this information in detriment of routing. Because of that, it is not recommended to use this protocol in MANs or WANs (Metropolitan or Wide Area Networks) like the ones we are studying. Another reason is that it is difficult to set up a measuring campaign: arranging this in a big network with hundreds of nodes, taking the CPU consumption into account, is a challenge.

- SNMP (Simple Network Management Protocol) [27] is a network protocol used for monitoring tasks. The better feature of SNMP is that, it is present in all network cards. That avoids us to have determined machines for measurement. Also it does not affect too much the routing task of a router as Netflow does, because it works with MIBs (Management Information Bases), which involve a very low computational cost. These databases store information where we can find, among others, the amount of traffic sent over a link (without taking into account the destination), with a typical granularity of five minutes. However, the problem is that we get the aggregated value for all the flows passing through the link, so an inference procedure has to be applied in order to differentiate the flows.

The second problem comes out mainly because commercial ISPs do not want to publish these data. It is understandable because enterprises try to compete but, on the other hand, it is the most solvable of the reasons that slows down the development in this field. Sometimes, researchers who want to get access to this data are allowed to capture samples in real time by the own organisations, but long-time datasets are pretty hard to obtain anyway.

In this aspect we must thank Zhang and Uhlig [1, 15] for publishing a couple of datasets for GÉANT and Abilene that we will use in our experiments, since they set our starting point for modelling research.

1.3.1. Inferring matrices

So, with SNMP we gather information about the traffic over the links, as with NetFlow we got information for each flow. With SNMP we do not have full information, so we have to infer something to complete the matrix.

This lack of real data can be solved by estimating the unavailable data. For this estimation we set the next expression:

$$y = A\,x \qquad \textbf{(1.1)}$$

where Y stands for the vector of link counts (of length K, where K stands for the amount of links in the network), A is the K x N^2 routing matrix (a_{ij}=1, if link i

belongs to the path associated to the origin-destination pair *j*) and *x* is the traffic matrix (written as a N^2 x *1* vector, x_{ij} is the traffic value associated with the origin-destination pair *i-j*).

Unfortunately since $K<<N^2$ deducing a traffic matrix from the SNMP measurement becomes an unsolvable inverse problem with infinite solutions, and therefore we need extra information in order to get a solution. One of the models which give us this additional information is the Gravity Model.

1.3.2. Gravity model

The Gravity Model [13] is based on the original Newton's gravity law, which states that the attraction force between two bodies is directly proportional the product of their masses and a gravitational constant, and inversely proportional to the square distance between them. This idea has also been used in social sciences and transport, where we find gravity models for economic trades or even migration flows. In all of these cases, the formulation of the model may lead us up to a nice analogy between original parameters for mass or distance and adapted parameters for each gravity law interpretation. The gravity model depends on the product of the masses (population, terminals) of two bodies (cities, nodes), a constant value (universal gravitational constant in original formulation) and distance (correlation, adjacency...). In case of migrations, people tend to migrate to bigger cities (the bigger the masses, the heavier attraction force); in case of economics, economic merchant giants will concentrate the business and trades.

In our case, the gravity model assumes the hypothesis that the traffic flows between two cities with high population will be denser than the flows of two cities with less population. As the gravitational force, the amount of data flowing between Madrid and Barcelona will be high, because there are more "people" (terminals) who use some service from ISPs. Alike, the flows between Castelldefels and Barcelona should be less intense than the ones between Barcelona and Madrid.

So, to compute the gravity model (tests and code in Annexes A.VI, A.VII and A.VIII) for a TM, we will not consider the physical distance (the original gravity law parameter of the distance is not quite evident yet in this model: routing algorithms under our network can redirect our packets through alternative ways, so the distance is not the physical distance), and we will use ingressing-egressing traffic volumes instead of using directly the population, so we will have 2N parameters (N for the fraction of total traffic generated by each node and N more for the fraction of total traffic received by each node). This amount of necessary data allows getting the real measurements from SNMP and estimating the remaining necessary data.

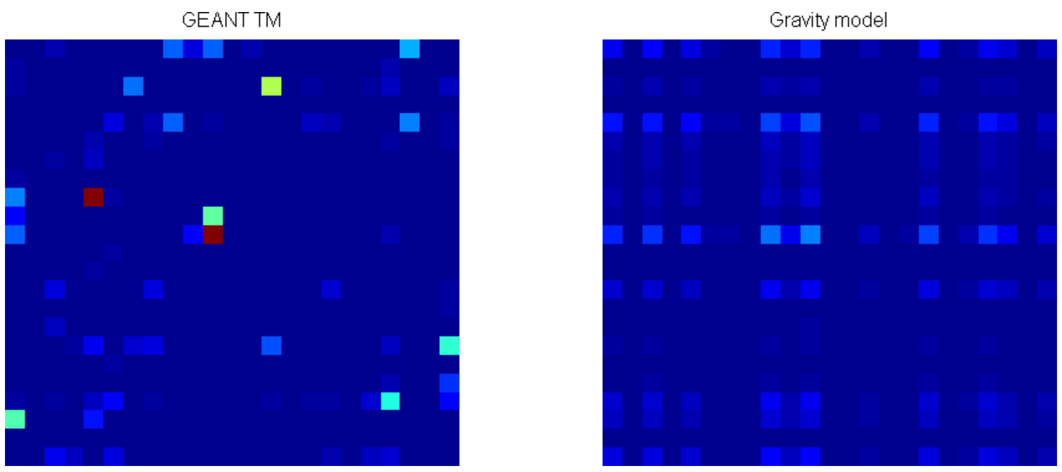


Fig. 1.3 Gravity model (GM). Left: TM from GÉANT (2005 February, 22nd, at 13:00). Right: Gravity model over the previous matrix. When obtaining the gravity matrix, the total traffic is conserved: we can then affirm that the energy is conserved. Mathematically $GM=T_i \times T_o^T \times TotalTraffic$ (where T_i is the entering fraction traffic vector and T_o is the exiting fraction traffic vector).

At this point we can introduce the concept of tomogravity to see how the inferred model is used in our TM inference scenario. It follows two steps: first, we get the Gravity model (which will be explained later) as an initial solution using edge link load data and ISP routing policy; then we make a tomographic estimation, refining the initial solution by applying some distance metric to find the solution (in the constraint subspace *y=Ax*) closest to the original model.

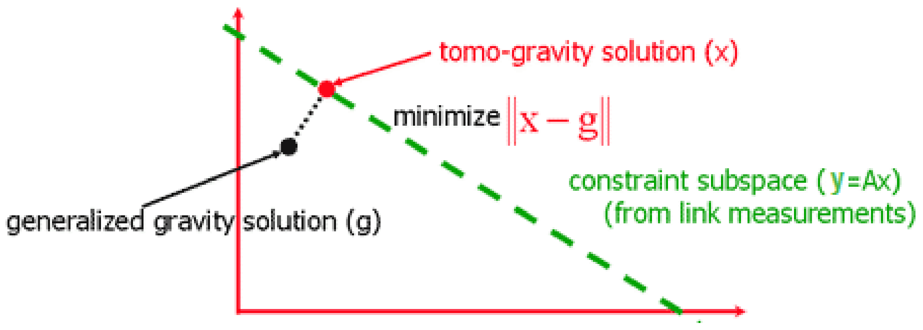

tomo-gravity solution (x)
minimize $\|x - g\|$
constraint subspace (y=Ax)
(from link measurements)
generalized gravity solution (g)

Fig. 1.4 Least square solution. We can see the variation from the gravity model to the constraint subspace defined by a single constraint equation [23].

Finally, note that in [13], the authors report that the method is good and provides a ±20% bound to the error (measured as mean square error, MSE, between the original TM and the one found with the tomogravity procedure).

CHAPTER 2. TOWARDS A MULTIRESOLUTION-BASED MODEL FOR TMs

2.1. Looking for a sparse model

One of our long-term goals is to develop models for TMs. As we have seen when we described the difficulties when measuring TMs, it is important to have good models. So, we want our model to be sparse, in the sense of having fewer parameters than the dimension of the matrix itself (i.e. having lots of “0” in the model). If we take an *N*-node network, the TM will have N^2 elements. In big networks, N can be of the order of hundreds or thousands, while our sparse model will only have $M<<N^2$ elements.

A sparse model will satisfy some interesting features [6]. For example, there is a trade off between the predictive capability of the model and its fidelity: with large amounts of coefficients we will get a very specific model which fits closely a specific set of data, but it will be less accurate with another sets; on the other hand, if we build our model with few data from different sets, we will achieve a predictive model for all the sets, but it will not fit them as closely (the errors and deviations will be higher). Likewise, if the model has few parameters, the compressibility of the data will be higher, so we will be able to refer to certain features (if any) with little computing power. It will also be easier to understand the physical meaning of a few values than thousands. Finally, we could get a better solution for the inference problem mentioned in 1.3.2: we have *K* measurements (where $K=O(N)$) and N^2 parameters to deduce, i.e. more unknowns than equations (this problem might become solvable if the model has $M \leq K$ values).

2.2. Wavelets and Multiresolution analysis (MRA)

The technique we use in order to obtain a sparse model is the multiresolution analysis. Multiresolution Analysis (MRA) allows us to analyze a dataset at different scales of detail.

Mathematically, MRA [29] consists in obtaining a set of nested subspaces of V_j (approximations) and their orthogonal complements, another set of nested subspaces of W_j (details), which verify that the difference between two consecutive approximation subspaces is exactly the detail of this scale ($W_j=V_{j-1}-V_j$). So for approximations V_j (for $j \geq 1$) there will always be the possibility to decompose this scale in the approximation and the detail subspace. In the inverse way we will be able to rebuild a scaled space if we have its approximation and detail subspaces.

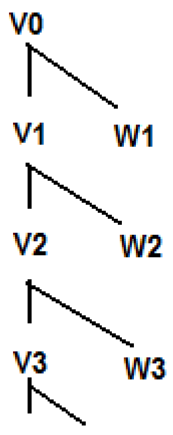


Fig. 2.1 Example of a MRA analysis. We see, at the beginning, the original signal is divided into details (W) and approximations (V). At the next step, these approximations will be decomposed again into details and approximations.

The wavelet transform [16,20] is one of the preferred techniques for developing a MRA. We can understand a wavelet as a mathematical function used to divide a signal into various scale components, i.e. the signal can be represented in translated and dilated versions of a basic waveform called mother wavelet.

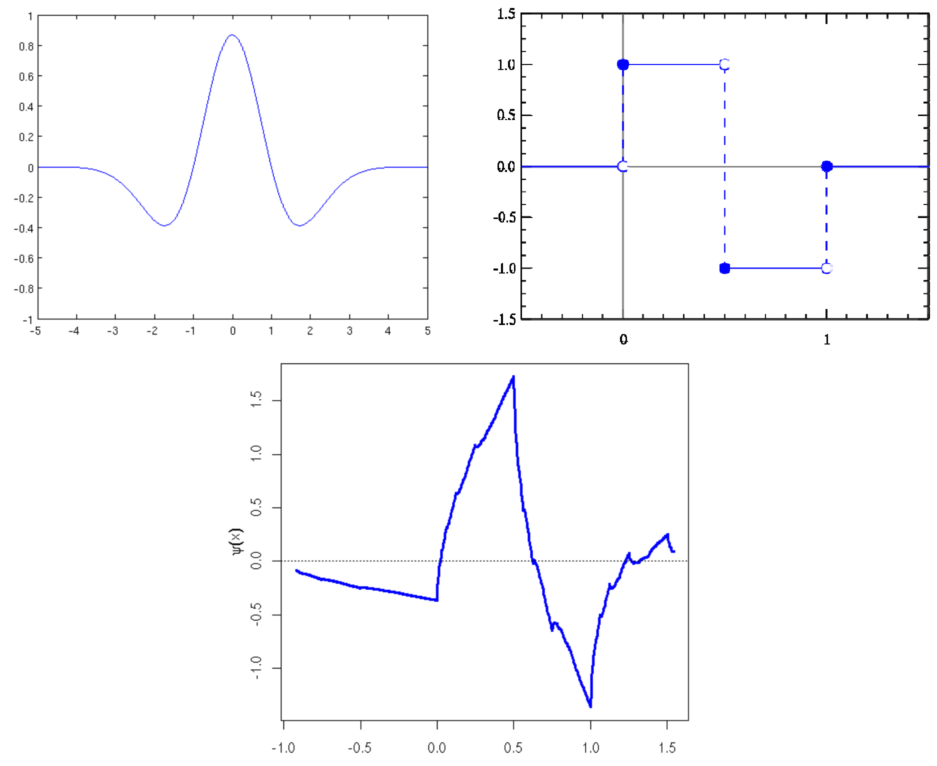


Fig. 2.2 Examples of mother wavelets: "*Mexican hat*", left; "*Haar*", right, "Daubechies'" down.

In signal processing it is usual to use wavelet-based methods to compress and denoise signals, time series or images [16]. The *Discrete Wavelet Transform* (DWT) [25] analyzes the signals from its scalar product with two base functions called father wavelet (or scaling function *φ(t)*) and mother wavelet (*ψ(t)*), with finite longitude. They are dilated (in powers of 2) and translated to cover the whole original signal's domain, obtaining the signal analysis at instants $t = 2^j - k$ (where *j* is the scale and *k* is the temporal displacement).

The role of the scaling function is to capture signal's low frequencies, while high frequencies (or details) are analyzed by the mother wavelet. If the base functions meet certain conditions, the resultant transform is orthonormal and can be easily implemented with a low and high pass filter bank (*h(n)* and *g(n)* respectively, and related with *φ(t)* and *ψ(t)*), as shown in figure 2.3.

In each step, low pass filters show us the successive approximation of the scales as they become less detailed. We can interpret it as an image which is progressively blurred as it passes through the low pass filters. On the other hand, in each iteration the high pass filters capture the high frequency details: $d_x(j,k)$ catches the difference between $a_x(j-1,k)$ and $a_x(j,k)$, where *j* is a lower (or more blurred) decomposition level than *j-1*. This method allows us to recover the original signal just repeating the process in the inverse way: synthesizing approximations and details starting from the wavelet transform coefficients.

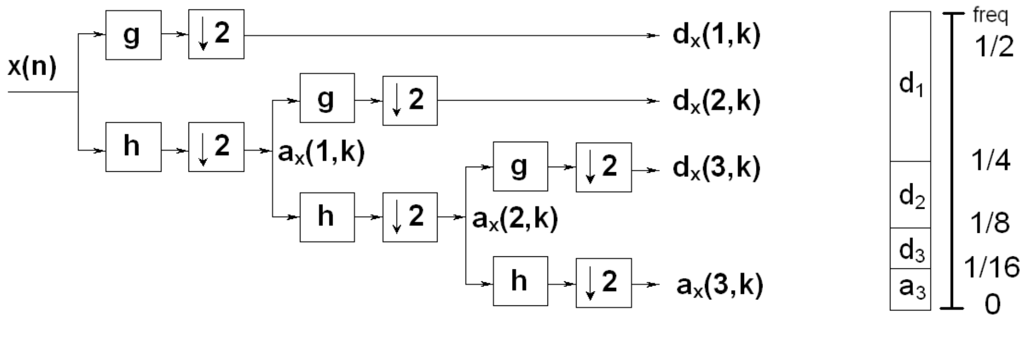


Fig. 2.3 Left: high-pass and low-pass filter bank for a DWT and *j*=3 scales, getting the V_3 approximation *a(3,k)* and the W_3 details *d(3,k)* for *j=[1,2,3]*. Right: Subsequent normalized spectrum decomposition.

In the frequency domain, DWT creates a subband decomposition whose spectrum is halved at each scale. This produces a multiresolution analysis where the original signal is decomposed in its final approximation (at the lowest frequency it will stand for the mean of the signal) and a set of details at different high frequencies (wavelet's coefficients). In the following subchapter we will see how to generalise this process to images in two dimensions.

In summary, the signal *x(n)* is decomposed in a first step with a pair of filters. The signal obtained at the output of the high-pass filter (*g*) is known as details d_1. The one obtained at the output of the low pass filter (*h*) is the approximation a_1. In a second step, we can decompose the approximation a_1 in its details d_2

and approximation a_2. The a_2 signal can be decomposed again in its approximations and details… and this process can be repeated as many times as we want. In addition, when we have decomposed the signal, we can compute the process in the inverse way to recover the original signal. If we have discarded some detail coefficients we will be able to rebuild an approximated signal.

2.2.1. Wavelets for 2D images

One of the fields on which wavelets have been an important contribution is in image processing and compressing. In JPEG 2000 [30], for example, wavelets are used to reduce the amount of bytes needed for coding the image, while maintaining an acceptable quality.

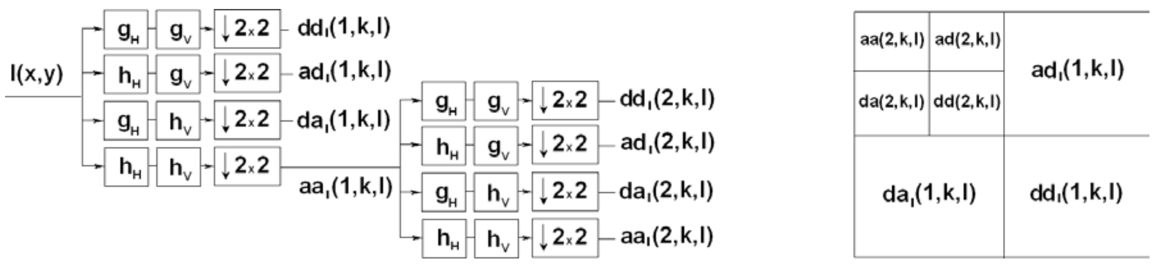


Fig. 2.4 At left, high-pass and low-pass filter bank used in 2D wavelet process. At right the decomposition of the normalized spectrum. Note the difference between 1D wavelet in Fig 2.2

2D wavelets are a generalization of classical wavelets, computed in both *x* and *y* axis of a bidimensional data. The way it works is similar to the procedure explained previously for signals in one dimension. The only thing we will have to consider is to splitting each subspace in four, each one corresponding to the four combinations of high pass and low pass filters (*h(n)-h(n); h(n)-g(n); g(n)-h(n); g(n)-g(n)*), as we are developing the previous process both in horizontal and vertical image's axis, so we will have four combinations of high and low pass filters, as shown in figure 2.4.

Let's see an example with a real image in Figure 2.5. In the decomposed image above we can see the approximation image in the little upper left square. In the rest of squares we can find the details at first scale (bigger squares) and at the second scale. In each case we have three outputs, which stand for the combinations of high-low, low-high, and high-high pass filters. In this image we notice that the details gather the silhouette of the building in the picture. If we were performing JPEG 2000, we could discard some coefficients of these details and keep the approximation, achieving some loss of quality (erasing the shape we get a loss of definition).

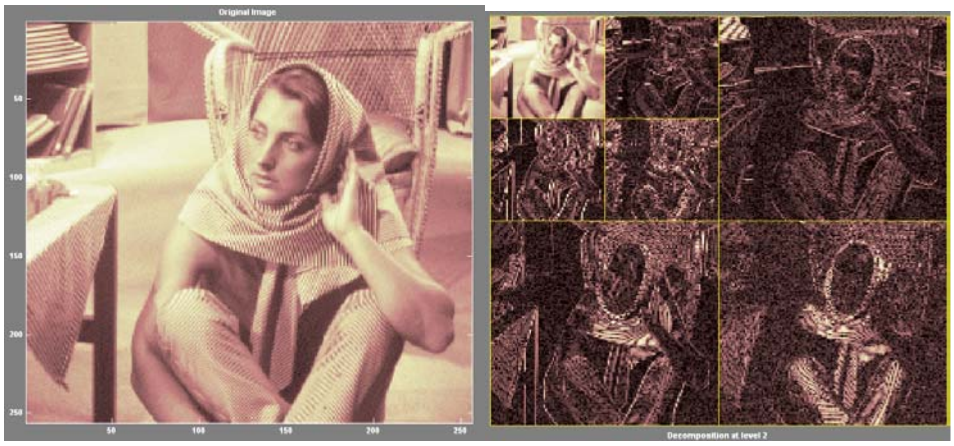


Fig. 2.5 Left: original image sample; Right: 2D wavelet decomposition of that image.

2.2.2. Diffusion wavelets

Our goal is to extend the multiresolution analysis to graphs. The problem is that we cannot apply the process for images with traffic matrices, since the relation between the nodes of a graph (a network with terminals or nodes and links between them) is not exactly the same than the one between the pixels of an image: a graph is more complex. In images (e.g., JPEG) part of the compression comes from inference: the value of a pixel can be recovered from the values of the nearer pixels. Obviously, if a certain pixel is black, we can assume with a high probability that the pixel at its right will be black or, at least, dark. But this relationship does not happen in traffic matrices.

And as we have seen before, TMs are defined over a graph, and the graph does not satisfy this condition of the pixels, even though we visualize the matrix as an image: two pixels of the image are adjacent but in the network they can represent nodes that are on two different extremes of the network. So we need a different tool to be able to apply the multiresolution analysis over a graph. To perform this analysis we need a different class of wavelets.

This first approximation, brought by Crovella and Kolaczyk [5] introduced graph wavelets as an extension to the wavelet transform in 2D. Graph wavelets allow computing the difference between link loads in nodes separated by a certain number of hops: the concept of scale is replaced by that of hop distance between links. The authors also show how graph wavelets can be used to anomaly detection. However, there is a lack of a fast computational algorithm, and the results of the transform are not orthonormal. In addition, graph wavelets do not show the traffic in a sparse mode: they give a redundant decomposition which is very similar to the result obtained with the continuous wavelet transform.

There is where another generalisation for wavelets comes into play. *Diffusion wavelets (DWs)* [6,9] allow us to develop the multiresolution analysis over a graph or a manifold. Intuitively speaking, a manifold is a mathematical space where two points of the same body can be very close in the Euclidean space, but the distance over the body's surface might be much longer.

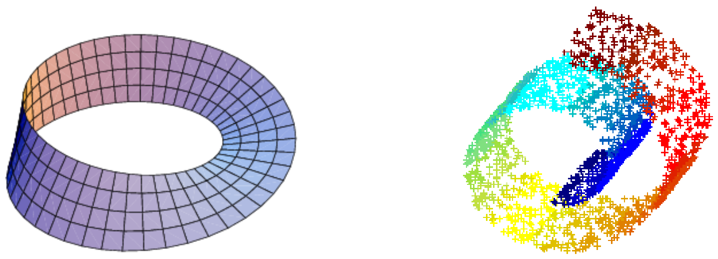


Fig. 2.6 *Möbius strip* (left) and *Swiss roll* (right): examples of manifolds. Two points are near in space but far along the body. We have to take into account if we are defining functions on a manifold because the interpretation of neighbourhood may vary due to the surface.

Diffusion Wavelets, developed by Coifman and Maggioni [9], have become an essential tool because they allow us to perform the MRA over a graph. First of all, we have to define the graph and *diffusion operator*, which play the same role as the scaling function on classic wavelets. This operator is defined by a matrix and can be understood as the base over which the function is projected, i.e. the backbone of the diffusion.

The idea behind the diffusion operator is to explore the graph (distance, adjacencies...), by means of the diffusion induced by itself. In the first experiments we have use a unweighted random walk as our operator. The random walk takes into account the adjacent nodes and gives each link a weight equal to the inverse of the number of links exiting the node. If we see the graph as a Markov chain, it will stand that the probability to "jump" across each link is the same (equiprobability). So we will obtain a symmetric and stochastic matrix for the operator. In terms of graph theory, the probability of escaping from a node across a certain link would be the inverse of the degree of the node. Notice that this operator does not catch the routing of the flows or other properties about the traffic distribution; it is just a first example. The next step is to dilate the operator by computing its powers: the n^{th} power of the random walk in terms of the associated Markov chain are the probabilities P_{ij} of going from a certain node i to node j in n steps. This progression gives up the final blurred result.

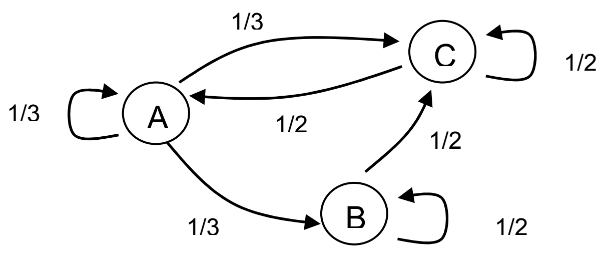


Fig. 2.7 Example of unweighted randomwalk in a network with three nodes.

We assume that a given function is defined in the graph vertices. Thus, when applied to the function, our operator will blur the given function over the graph. The result is that nearer nodes will mix and blur more quickly than further nodes.

As we are working over a graph instead of an image (we are developing the MRA), the analogous result of the DWT decomposition will be a subband selection in terms of the diffusion operator of the eigenvalues and eigenvectors. It is well known [19,24] that a linear operator T can be expressed as:

$$T = \sum_{i=1} \lambda_i v_i^T v_i \qquad \textbf{(2.1)}$$

where λ_i are the eigenvalues of T and v_i its associated eigenvector. We use a normalized version of T, so the highest eigenvalue will be one. We also have other interesting properties, for example: λ^n are the eigenvalues for T^n and as $|\lambda|<1$ the growing of n will make $|\lambda|$ tend to zero. The eigenvalues are related with the frequencies included in each subband. On the other hand, we have an extra parameter ε that will determine the precision of the decomposition. The idea is to classify the eigenvalues by comparing them to ε, so that eigenvalues lower than ε will be identified as details, while higher ones will be the approximations. This process will be repeated step by step until all but one of the eigenvalues fall below this threshold set by ε. [10].

Recall that the eigenvalues will be the solutions of equation:

$$(A - \lambda I)\, x = 0 \qquad \textbf{(2.2)}$$

where A is the matrix of the diffusion operator, I is the identity matrix, λ is an eigenvalue and x is its associated eigenvector. For a n x n diffusion operator matrix, we will have n solutions (n eigenvalues) and we can infer some things about the graphs from their eigenvalues. For example, the number of zero eigenvalues is the number of connected components in the graph, λ_1 is called the algebraic connectivity... These eigenvalues are useful to determine graph families, too [6,19,24].

This expression can be applied to the spectral theorem [24] because under specific conditions a linear transformation of a vector *v* can be expressed as a combination of its eigenvectors (v_i) and its eigenvalues (λ_i):

$$T(v)=\lambda_1(v_1 \cdot v)v_1 + \lambda_2(v_2 \cdot v)v_2 + \ldots + \lambda_{n-1}(v_{n-1} \cdot v)v_{n-1} \qquad \textbf{(2.3)}$$

After having normalized the operator by its higher eigenvalue, when we take the n^{th} power of *T* (T^n), *n* tending to ∞, every eigenvalue will be zero, except the higher one (which is 1, since the eigenvalues are normalized). Then we will only have an eigenvalue over the threshold ε. This way, and applying the diffusion operator once and again, we will get a divided spectrum of the graph: then, the kept eigenvalues will expand their low frequency subspace (approximation) and discarded ones will expand their high frequency subspace (details). At each step, kept eigenvectors are orthonormalized again.

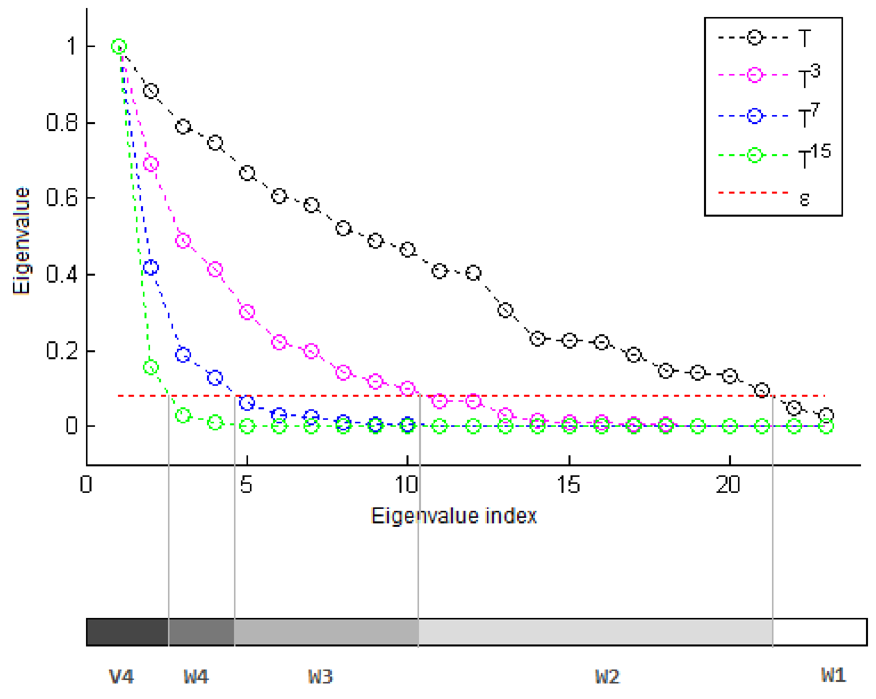


Fig. 2.8 Eigenspectrum of the powers of an operator *T*. Notice that as *n* grows more eigenvalues are discarded, i.e. assigned to a detail subspace W_i . When $n \rightarrow \infty$, only the biggest eigenvalue (1) will remain.

Once we have done approximation and detail subspaces (V_i and W_i) we can project any function *F(v)* (defined on the vertices of the graph) over the subspaces and get a MRA of *F(v)*. This process may have a high computational cost, but Coifman and Maggioni (see [9]) developed a fast algorithm to compute it and get the detail and approximation coefficients.

2.2.3. Previous work: Diffusion Wavelets in 2D

We know that traffic matrices can be represented as bidimensional functions *F(x,y)=z* where *x* and *y* are the ingress and egress nodes in this order and *z* stands for the traffic volume. Since the Diffusion Wavelet transform can only be applied to one-dimensional functions *F(x)*, we have to extend Diffusion Wavelets to 2D and we follow the same approach of the DWT in 2D [6]. We have already seen in 2.2.1 the process for extending the DWT to 2D: decomposition in four subbands (due to the two dimensions) for each scale and repeating the process in the next scale for the approximation subspace (result of the low-low pass filters). In general terms, it is the same process, and now we will get three detail subbands and only one approximation. Analogously, the DW in 2D transform still projects *F(x,y)* one time in each "dimension" (origin and destination) by taking the tensor product of the 1D base for both approximation and details. The algorithm details vary, but the process is intuitively the same. In this case we will call C_{vv1}, C_{vw1}, C_{wv1} and C_{ww1} to the transform coefficients which stand for VV_1, VW_1, WV_1 and WW_1 (where *VV* is the low-low frequency approximation subspace, *WW* is the high-high frequency detail subspace, and VW and WV are the low-high and high-low combinations). The 2D transform is also orthonormal and invertible, fact that allows us to rebuild the original function starting from its coefficients. In figure 2.9 we can see how we read the decomposition taking into account three subbands (diffusion wavelet in two dimensions) instead of four (two dimension wavelets for images).

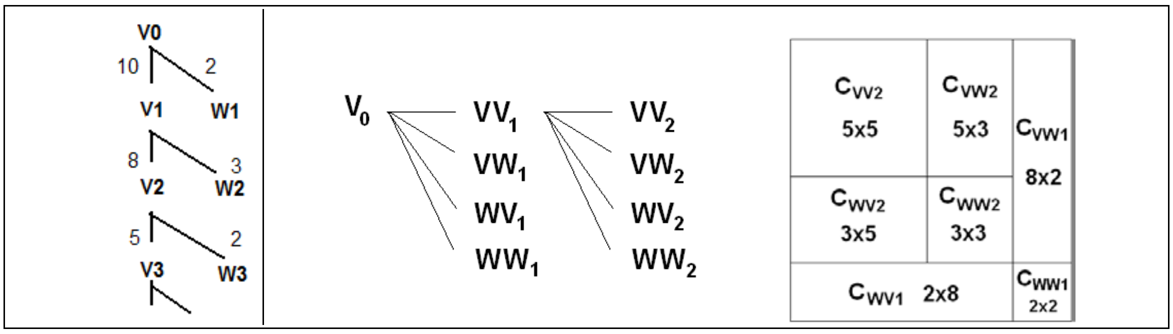


Fig. 2.9 Right: decomposition tree (defined subspaces and eigenvalues) for diffusion wavelets in two dimensions. Comparison with the decomposition for one dimension (left), with 10 eigenvalues. The dimensions (*n* x *m*) on the right side stand for the eigenvalues (λ).

In our first experiments with the DW in 2D we have studied more than 20000 traffic matrices from both Abilene and GÉANT. We looked for two main goals: see how the diffusion affects a traffic matrix in order to analyse its subsequent multiresolution (checking afterwards the capability of reconstruction, repeating the process inversely) and see the compressibility obtained with this method. We used the normalized unweighted random walk as our operator, and the precision has been $\varepsilon=10^{-7}$. We can see the MRA representation of a specific Abilene TM in Figure 2.11. The approximations are the VV subspaces (now renamed V, while the details W are the sum of WW, VW and WV). In the two first steps (*V_1* and *V_2*) the reconstruction is exactly the original TM because *W_1* and *W_2* have no eigenvalues, so we have not included the images in the figure. The effect of low frequencies can be easily appreciated as we reiteratively apply the operator in approximations, as well as the effect of high frequencies in details.

On the other hand, to check the compressibility of the TM we have made different tests taking long time data sets (from half a month to a whole month, as long as our public datasets allowed) in order to find how much energy from original matrices keeps concentrated in the first coefficients. We can see the results in Table 2.1.

Table 2.1. Energy preservation in coefficients in two representative sets (8640 TMs for Abilene, and 2880 TMs for GÉANT).

	Preserved Energy	
Set	80%	90%
Abilene, June 2004	4 coefs. (2.8%)	21 coefs. (14.6%)
GÉANT, March 2005	24 coefs. (4.5%)	75 coefs. (14.2%)

We can see in figure 2.11 how the image gets blurred as we advance in our diffusion process (each time analyzing more little subbands). As we have said, this is what we obtain when the diffusion is performed over the network.

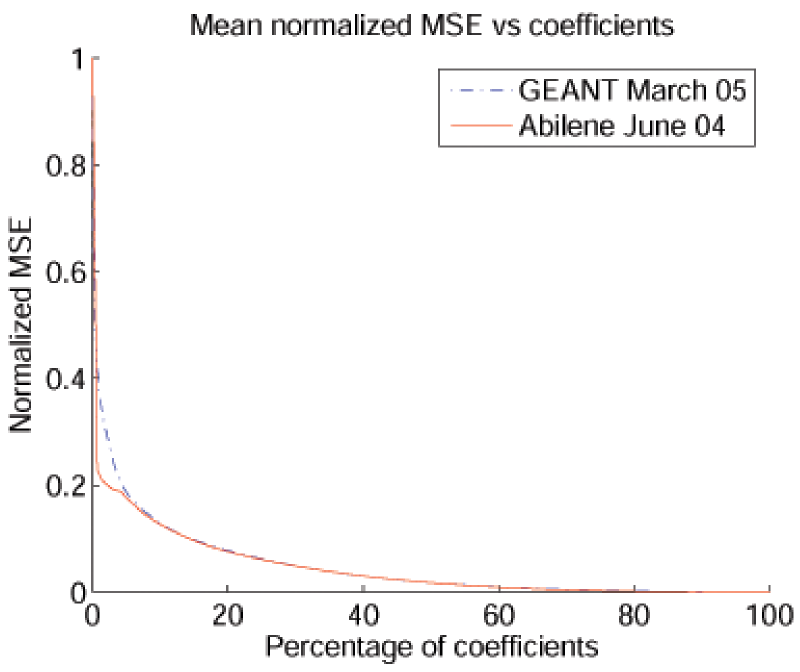
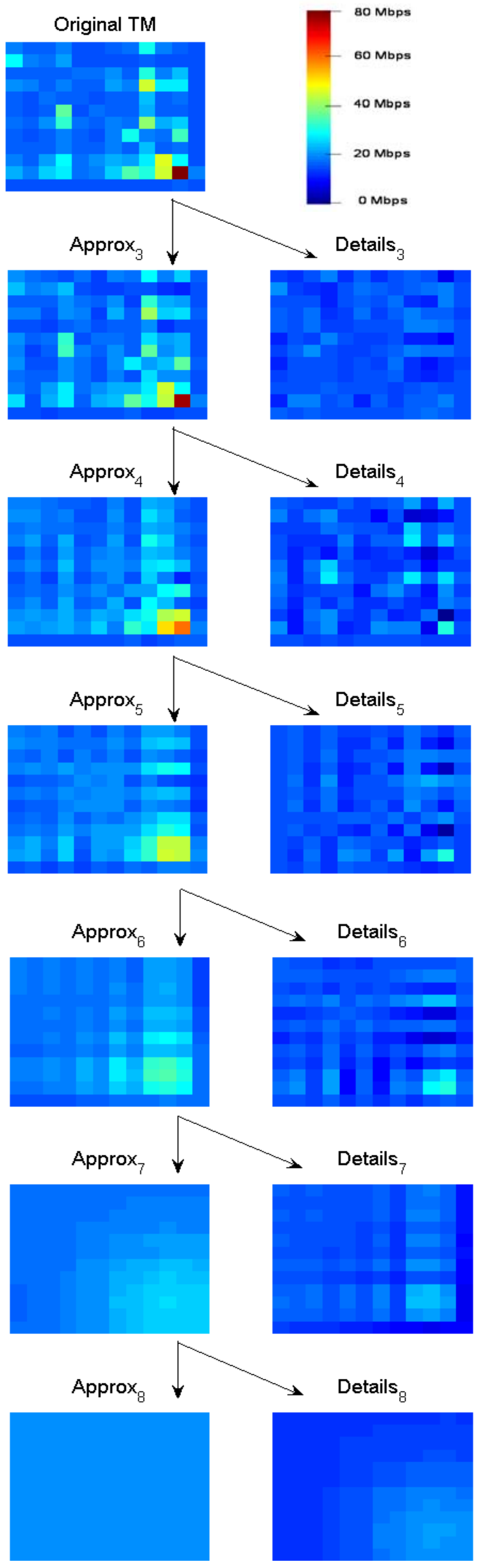


Fig. 2.10 Normalized MSE depending on the percentage of DW coefficients, for a monthly set with the unweighted adjacency random walk diffusion operator.



Eigenvalues	
Approx$_1$	0
Details$_1$	0
Approx$_2$	0
Details$_2$	0
Approx$_3$	10
Details$_3$	2
Approx$_4$	6
Details$_4$	4
Approx$_5$	5
Details$_5$	1
Approx$_6$	3
Details$_6$	2
Approx$_7$	2
Details$_7$	1
Approx$_8$	1
Details$_8$	1

Fig. 2.11 Decomposition: approximation and details subspaces and associated eigenvalues for the multiresolution analysis of the mentioned sample.

The results obtained with other sets of temporal data fit with the results of the table above and confirm the dispersion of DW representation: in mean, 15% of the coefficients gather 90% of the TM original energy. Figure 2.10 complements this analysis showing the Mean Square Error (MSE) of the reconstructed TMs taking into account the percentage of used coefficients in the reconstruction process. We must notice that, despite the difference between Abilene and GÉANT's datasets, the results are quite similar.

Finally we have noticed that each matrix has an own feature (like a signature) related with the order of the coefficients (depending on its energy contribution) which is, moreover, persistent in time. Figure 2.12 shows the relative order (the biggest coefficient has the darkest colour) of the biggest 20 coefficients from 14 days in the March 2004 dataset from Abilene. At, approximately, 3500th matrix, we can appreciate a structural change that changes this signature. When we go back to the original TMs (Figure 2.13) we can see a clear structural change in the 14th day (node 6 starts receiving a lot of traffic it didn't receive in the previous days, while node 9 decreases a lot its received traffic) .

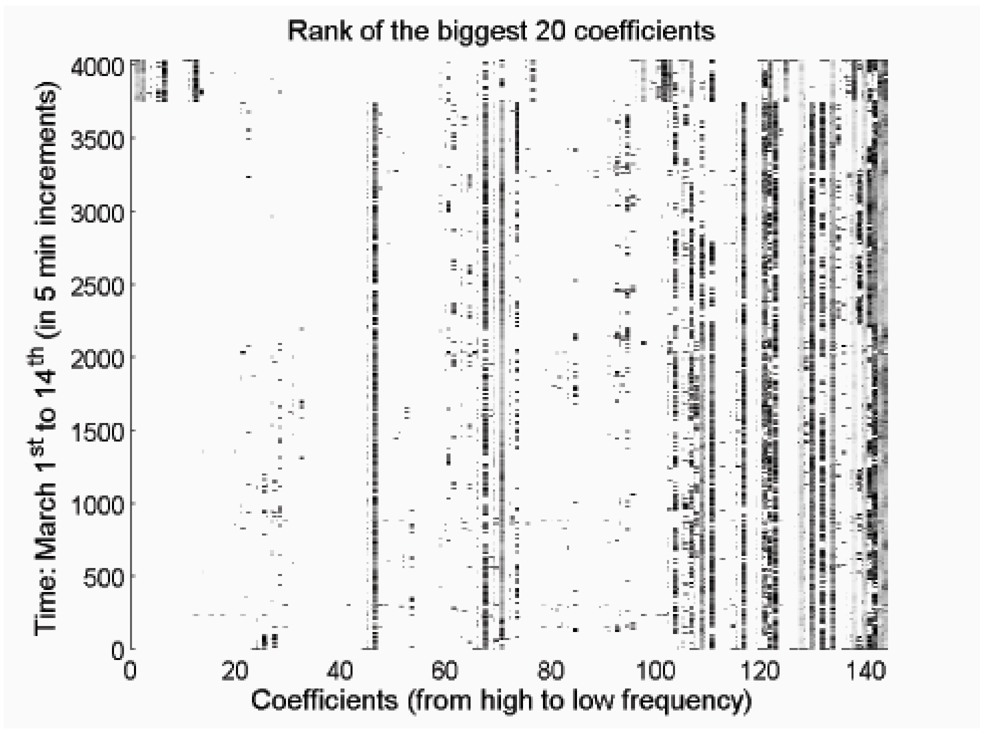


Fig. 2.12 Particular signature for 2004, March, days 1-14, from Abilene.

These changes can be measured just taking a few coefficients (if their energy is significant of the totality). This is far from turning into an anomaly detection algorithm, but the DW transform seems potentially useful for this purpose.

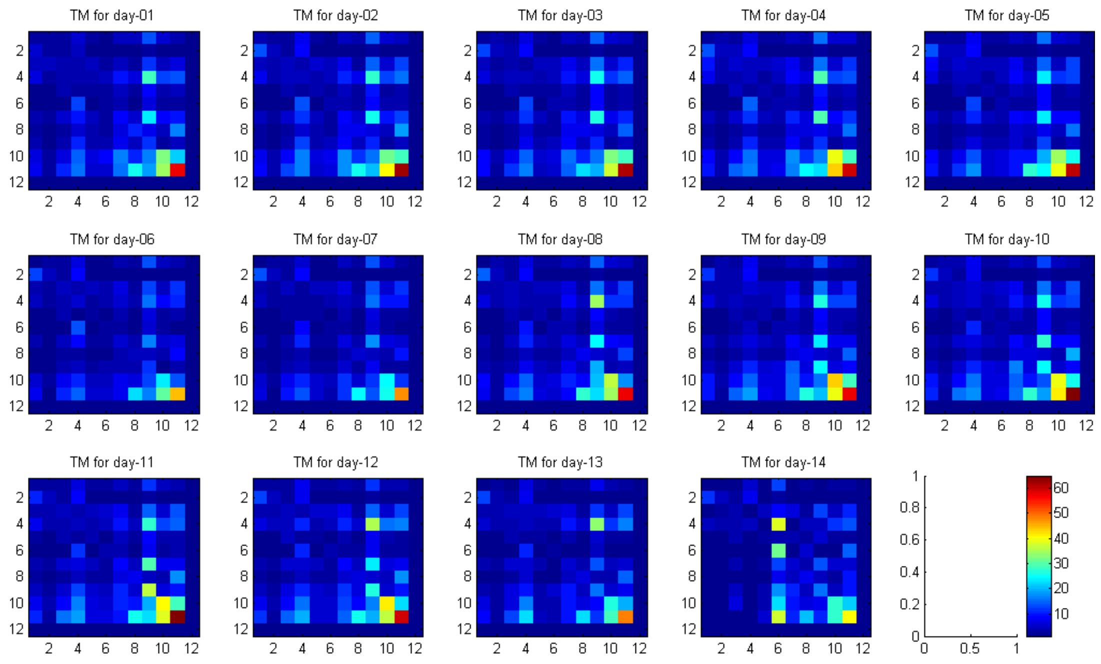


Fig 2.13 Daily mean values for TMs for the first 14 days of March 2004, Abilene. Notice the structural change on the 14th day.

As we have seen, the election of the diffusion operator will be an important factor in our procedure, as it defines the base over which the function to be studied (the TM, in our case) will be projected. The results for the random walk have been presented as an example, as it is the simplest operator, but the goal is to find a good operator. For this purpose, and again, as an example, we present another diffusion operator based on the gravity model. The idea is, in spite of working over the topology graph, we look at the real network and define another graph whose links are weighted according to the gravity model, i.e. nodes exchanging higher amounts of traffic became closer “neighbours” and those which exchange low volumes become “further”. Remember that the gravity model can be easily inferred by SNMP counters and so it is already available by direct measurement. Figure 2.14 shows a comparison between the topology/random walk operator and the gravity-model-based operator.

These results have been presented in [10], and through the development of this thesis have been repeated, so we can confirm their reliability. The results show an improvement in the compressibility when compared with the unweighted adjacency (random walk).

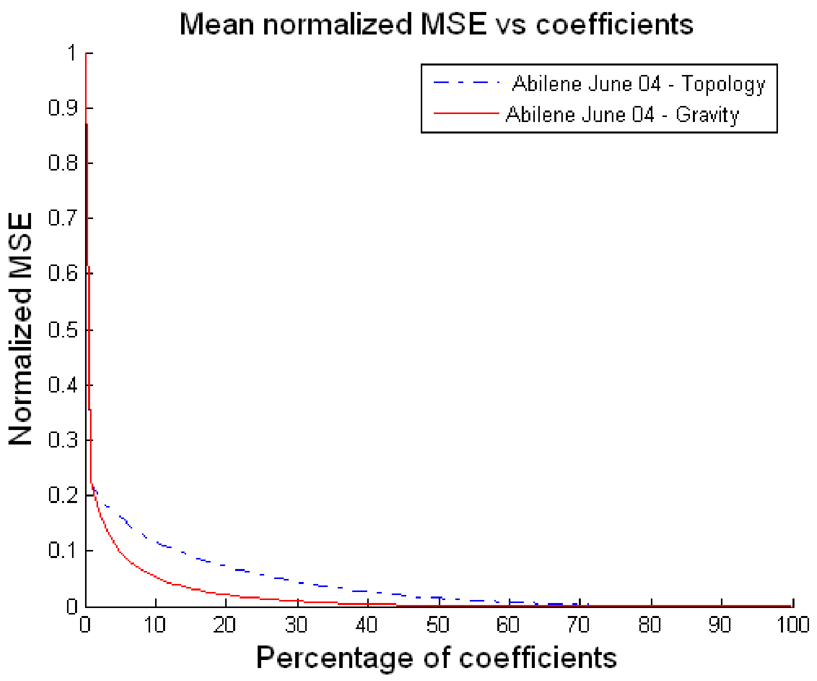


Fig. 2.14 Normalized MSE respect the coefficient percentages for two different operators in Abilene.

CHAPTER 3. EXPERIMENTS

3.1. Description of the dataset

After having described the details of our algorithms, the reader may be interested in being introduced to the datasets we have used for our experiments. In this thesis we have worked with data obtained from the north-American (Abilene [32]) and European (GÉANT [33]) academic networks. It must be noticed that the organisations may avoid giving their network information publicly: they may not want this information to fall into the wrong hands. So researchers have to ask specially for this data. Both Abilene and GÈANT came through with some solutions:

- Abilene chose to publish their data and network maps, linking each node to a Point of Presence, but protecting these data because it was obsolete and temporally deviated: the datasets we worked with are supposed to be from 2004, but we are not sure they were from 2004.

- GÈANT hided some information, besides of publishing old data. Their data is anonymized: we do not know which nodes correspond with each PoP, and, furthermore, we can find that samples do not agree with their names (i.e. "IntraTM-2005-01-01-00-30" does not have to be mandatorily from January 1st at 00:30 hours at night).

Abilene's samples are taken every 5 minutes, so we have a total of 12 matrices for each hour. In case of GÉANT we only have 1 matrix each 15 minutes (4 per hour). Despite this, we worked preferably with Abilene, because certain computations increase its time proportionally to the amount of nodes (for certain processes it is better to work with twelve 12x12 matrices per hour than with four 23x23 matrices per hour in the case of GÉANT). In some cases the computing capacity that Abilene's matrices will demand is affordable while with GÉANT's TMs it will not work.

It must be said that Abilene and GÉANT datasets are incomplete. Sometimes it is only one matrix which is missing, but there can be whole hours or days. For this reason, results shown here are from large continuous packs of data. With GÈANT data we have chosen 20 days in February 2005, which is one of the largest and continuous sets. In case of Abilene we will work with a full month, June 2004, where we will be able to see, apart from the day-night cycle, the week cycle. We usually refer to these cycles because it is an easy way to see if the sets are loaded appropriately; traffic values at night are lower than in day, it uses to be an easily-identification for the peak hour and traffic values at weekend days are poorer, too.

3.2. First steps

The first part of this research was to get used with the sets of data. For this purpose we introduce ourselves to the software: TOTEM [22] and Matlab [21]. However, we have not used TOTEM further than for visualizing topologies and play a little bit with the TMs. It has been useful to understand traffic matrices and see how the bandwidths can be redirected depending on the capacity of the nodes or its failures (reconfiguration of networks). MATLAB has been a little harder to understand, and we still discover functions or features that make the whole thing easier. In this chapter we will see some features, but for more information, see [8].

Following this, to check the right configuration of the software, we repeated some of the old experiments and reviewed the existing developed code to understand it better. And then it came the part where new ideas (the time-correlated operator) were tested.

3.2.1. Parsing our datasets

Once we have got our datasets, we need to parse it in order to visualize or work with the matrices with TOTEM. Traffic matrices come in XML format, either labelled or not, depending on if they are Abilene's or GÈANT's. Although the networks are different, the tagging is quite similar.

```
<?xml version="1.0" encoding="ISO-8859-1"
standalone="yes"?>
<TrafficMatrixFile […]>
    <IntraTM ASID="11537">
        <src id="ATLA-M5">
            <dst id="ATLA-M5">26.6666666666667</dst>
            <dst id="ATLAng">522.208</dst>
            <dst id="CHINng">1641.33866666667</dst>
            <dst id="DNVRng">335.728</dst>
            <dst id="HSTNng">413.032</dst>
            <dst id="IPLSng">489.874666666667</dst>
            <dst id="KSCYng">365.077333333333</dst>
            <dst id="LOSAng">817.869333333333</dst>
            <dst id="NYCMng">452.061333333333</dst>
            <dst id="SNVAng">747.405333333333</dst>
            <dst id="STTLng">388.317333333333</dst>
            <dst id="WASHng">3141.64</dst>
        </src>
[…]
    </IntraTM>
</TrafficMatrixFile>
```

```
<?xml version="1.0" encoding="UTF-8"
standalone="yes"?>
<TrafficMatrixFile […]>
  <info>   […]  </info>
  <IntraTM ASID="20965">
    <src id="12">
      <dst id="12">302691.3422</dst>
      <dst id="13">30623.7689</dst>
      <dst id="19">9156.3289</dst>
      <dst id="23">2726.5867</dst>
      <dst id="8">5659.2267</dst>
      <dst id="18">4138.5511</dst>
      <dst id="4">152568.9067</dst>
      <dst id="1">8988.0622</dst>
      <dst id="5">4168.9600</dst>
      <dst id="3">15538.8000</dst>
      <dst id="22">17130.4267</dst>
      <dst id="7">2514.1689</dst>
      <dst id="2">101214.7556</dst>
      <dst id="6">18.0089</dst>
      <dst id="16">8901.9911</dst>
      <dst id="14">434.1511</dst>
      <dst id="20">10964.5867</dst>
      <dst id="11">4511.0489</dst>
      <dst id="9">4441.8667</dst>
      <dst id="17">86795.6356</dst>
      <dst id="21">13459.1644</dst>
      <dst id="15">3841.1556</dst>
    </src>
[…]
  </IntraTM>
</TrafficMatrixFile>
```

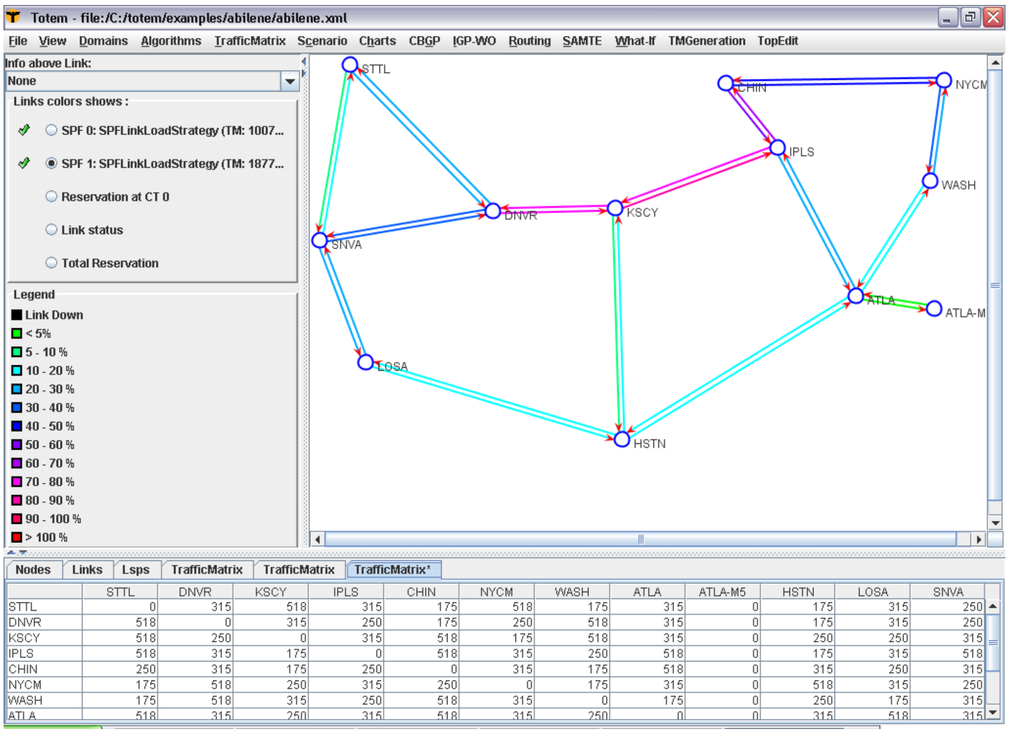
Fig. 3.1 Two samples of Abilene's (left) and GÈANT's (right) TMs in XML format.

The application we have used for parsing was TOTEM [2,22]. TOTEM (TOolbox for Traffic Engineering Methods) is a project to develop a toolbox of algorithms for traffic engineering. It is mainly developed by the Université Catholique de Louvain in collaboration with the Université de Liège.

TOTEM is an application that will help us at the time of analysing topologies, also featuring the capability of assigning some traffic matrices to these topologies. Here we can see an example of its utilization for these purposes. As a curiosity, some of the traffic engineering algorithms only run in LINUX.

We start by loading some topology (when getting the datasets from Abilene and GÈANT, there is also a XML topology file) in a very useful graphical interface, from which we may gather some extra information only passing the mouse over the links or nodes. Once we have the topology loaded, we only will have to load a traffic matrix, set a routing algorithm, extra traffic engineering features (if any) and run the scenario. We will see on the GUI (*Graphical User Interface*) the graph according to the network we loaded. The links will be coloured depending on their utilisation, and we will be able to configure bandwidths, to set up or down links, or to implement routing patterns in order to see the behaviour of the network.

TOTEM works routing traffic matrices over the loaded topology. It performs the computing and distributing of the loads over the links. Since we can modify the topology, the algorithm takes into account the changes made to compute again the loads if necessary. Additionally, we can plot charts to have an easier view of comparisons between algorithms or configurations.


Totem - file:/C:/totem/examples/abilene/abilene.xml
File View Domains Algorithms TrafficMatrix Scenario Charts CBGP IGP-WO Routing SAMTE What-If TMGeneration TopEdit
STTL
CHIN
NYCM
IPLS
WASH
DNVR
KSCY
SNVA
ATLA
ATLA-M
LOSA
HSTN
Nodes Links Lsps TrafficMatrix TrafficMatrix TrafficMatrix*

Info above Link: None

Links colors shows :

✓ ○ SPF 0: SPFLinkLoadStrategy (TM: 1007...

✓ ● SPF 1: SPFLinkLoadStrategy (TM: 1877...

○ Reservation at CT 0

○ Link status

○ Total Reservation

Legend

■ Link Down
🟩 < 5%
🟩 5 - 10 %
🟦 10 - 20 %
🟦 20 - 30 %
🟦 30 - 40 %
🟦 40 - 50 %
🟪 50 - 60 %
🟪 60 - 70 %
🟪 70 - 80 %
🟥 80 - 90 %
🟥 90 - 100 %
🟥 > 100 %

	STTL	DNVR	KSCY	IPLS	CHIN	NYCM	WASH	ATLA	ATLA-M5	HSTN	LOSA	SNVA
STTL	0	315	518	315	175	518	175	315	0	175	315	250
DNVR	518	0	315	250	175	250	518	315	0	175	315	250
KSCY	518	250	0	315	518	175	518	315	0	250	250	315
IPLS	518	315	175	0	518	315	250	518	0	175	315	518
CHIN	250	315	175	250	0	315	175	518	0	315	250	315
NYCM	175	518	250	315	250	0	175	315	0	518	315	250
WASH	175	518	315	250	518	315	0	175	0	250	175	315
ATLA	518	315	250	315	518	315	250	0	0	315	518	315

Fig. 3.2 TOTEM screenshot example, with its GUI. We can see the topology of Abilene and the coloured links show the utilisation over them.

In TOTEM we can load different matrices at time. In the next figure, we can see the chart for the matrices in the first day of the Abilene dataset (2004 March, 1st from 00:00 to 23:55, with samples each five minutes).

In conclusion, TOTEM gives us a very visual way to understand the routing algorithms, working with TMs. It allows us to simulate network scenarios, too. For an extended tutorial introduction for TOTEM, see Annex A.I.

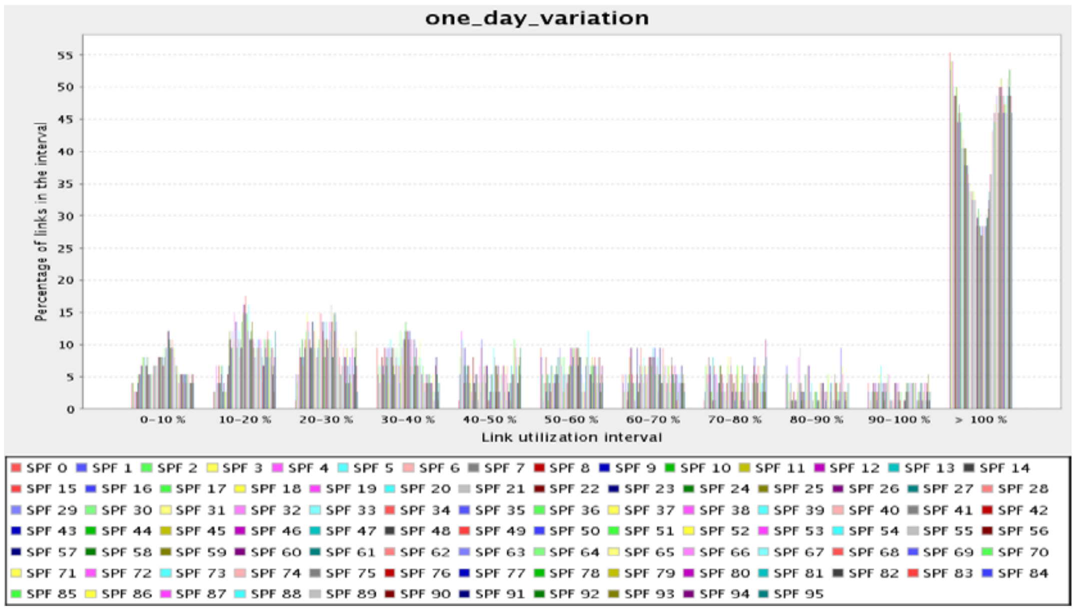


Fig. 3.3 First day's matrices. Each set of bars deals with one range of utilisation. We can infer, by displaying the whole set, the day-night cycle.

3.2.2. Visualizing TMs in Matlab

In order to analyze statistically the matrices, we need other tools, such as Matlab. Matlab is developed by Mathworks, and offers a great collection of functionalities related with many mathematic fields.

With Matlab, one interesting point will be to visualize the matrices. For this purpose, some code has been developed in order to parse TMs so we can display them. If we work with them as plain numbers it will be more difficult to see their differences and to detect neither patterns nor anomalies. In the figure 3.4 we will see different kinds of visual representation for a traffic matrix. The matrix represented is TM-2004-06-01-1300 (June, 1st at 13:00).

First of all, we have the *plot* function. It draws each row of the matrix as an independent series, assigns each series a colour (see legend) and plots all series. In this case, where we are just visualizing one matrix it is not an interesting graphic, but when we look at hundreds of matrices plotted at the same figure, we can see some behaviour patterns, or we can infer where will be the mean and standard deviation.

Secondly, we have *imagesc* function (see figures 1.1 and 1.2 for extra examples). This function turns the matrix into an image, assigning each “pixel” (understanding pixel as every [i,j] position in the matrix) a colour depending on its traffic value. With this representation we can see which the hottest flows on the network are. The main difficulty is that we can only visualize one matrix at the same time. However, we can use methods to combine matrices in order to see them at the same image.

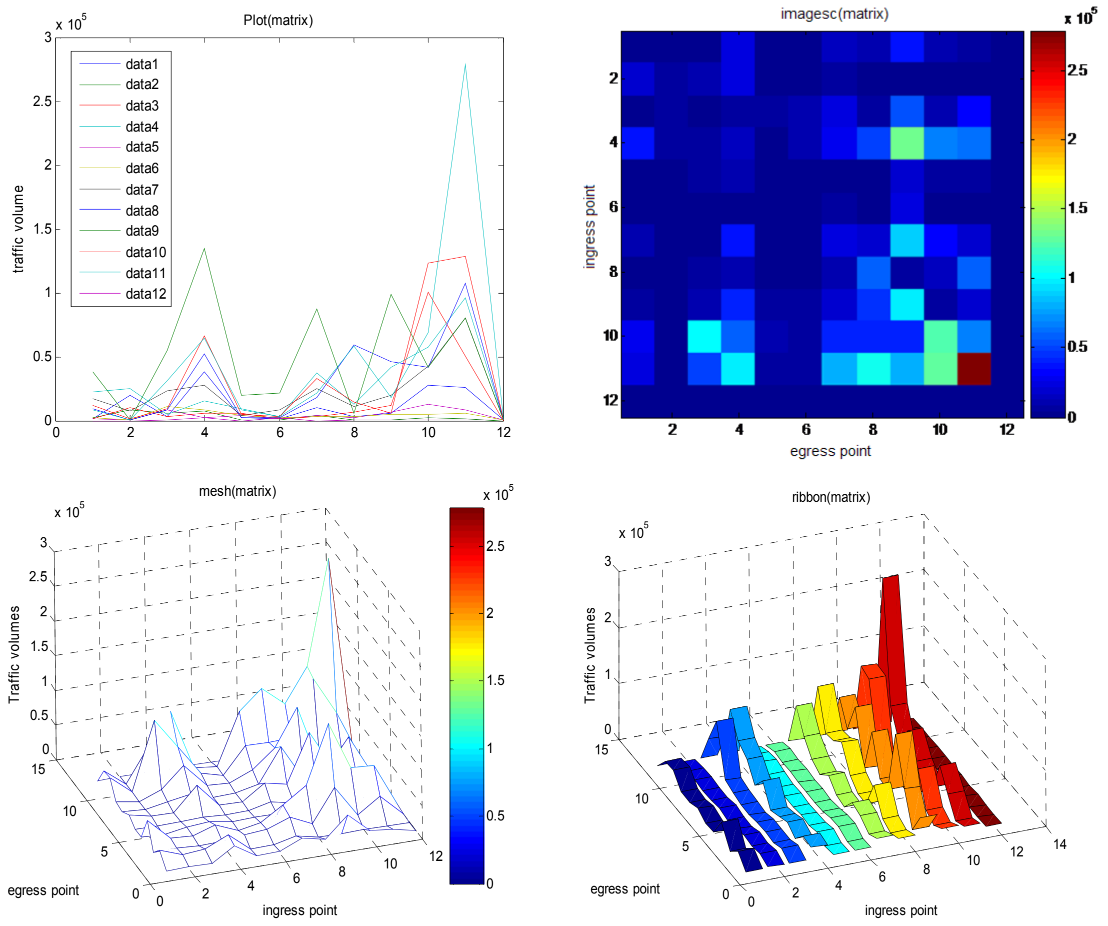


Fig. 3.4 Plot – useless in this case – (up, left), imagesc (up, right), mesh (down, left) and ribbon (down, right) representation functions.

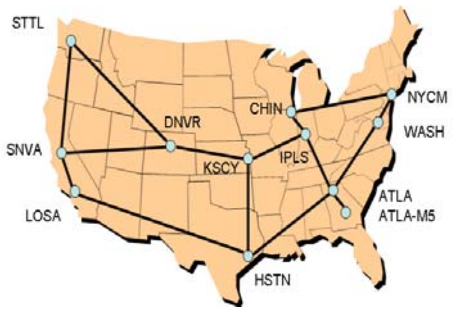
Third, we have the *mesh* function, which allows us to draw the matrix in three dimensions. In 2D representations we had to sacrifice one variable (mainly in *plot*, where egress nodes turned into different colour for series); *imagesc* achieved a better mode to see three variables in a 2D graph. Now, we can see the matrix in 3D and isolate each variable from the others: what is more, we can rotate the image until we see it in 2D (only two variables), and watch the projection of one axis over the rest.

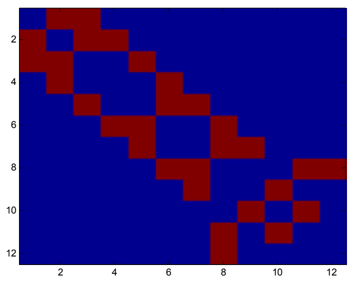
Finally we have the *ribbon* function. It is very similar to *mesh*, but ribbon shows us the series in 3D (i.e. like a *plot* in 3D). This lets us to compare series and watch their fluctuations (this was especially difficult with *plot* if we were visualizing lots of matrices). For more visualizations see Annex A.II.

3.2.3. Other matrices in our study

Apart from traffic matrices, we also need to represent other matrices. In our research we will also use two other matrices, namely the *adjacency* and *Laplacian* matrices (tests in Annex A.IX).

The adjacency matrix *B* is defined as follows: $b_{i,j}$=1 means there is a direct link between *i* and *j* nodes, $b_{i,j}$=0 means that *i* and *j* are not linked directly. This way, this binary matrix, tells us the neighbourhood relationship between the nodes in the network or graph and will be very useful when we are testing diffusion operators.





```
0 1 1 0 0 0 0 0 0 0 0 0
1 0 1 1 0 0 0 0 0 0 0 0
1 1 0 0 1 0 0 0 0 0 0 0
0 1 0 0 0 1 0 0 0 0 0 0
0 0 1 0 0 1 1 0 0 0 0 0
0 0 0 1 1 0 0 1 0 0 0 0
0 0 0 0 1 0 0 1 1 0 0 0
0 0 0 0 0 1 1 0 0 0 1 1
0 0 0 0 0 0 1 0 0 1 0 0
0 0 0 0 0 0 0 0 1 0 1 0
0 0 0 0 0 0 01 0 1 0 0
0 0 0 0 0 0 0 1 0 0 0 0
```

Fig. 3.5 Relationship between Abilene's network (graph) and its adjacency matrix.

In the figure we find the representation of Abilene's adjacency matrix and its value. The analogy is visible: red pixels are one and blue pixels are 0. Each value [i,j] stands for a node combination: the first 0 is the adjacency of Seattle to Seattle (it is 0 because it cannot be its own neighbour). Then, if we observe the first line, we will find Seattle's neighbour nodes in this order: Sunnyvale and Denver (the order is given in fig 1.1, Abilene). Notice that in this case the diagonal is zero (the adjacency of each node to itself) although in our case we will also consider the traffic flowing from one node to itself (and therefore we will add an identity matrix).

There is a variation of this matrix that is also useful. It is the *Laplacian matrix* or *admittance matrix* of the adjacency. The Laplacian matrix *L*, is defined over a graph as follows: $l_{i,j}$ = -1 if *i* and *j* are neighbours (vertices v_i and v_j linked by an edge), $l_{i,j}$ =*deg*(v_i) only when *j* = *i*, and $l_{i,j}$ = 0 otherwise. The Laplacian matrix shows us the neighbourhood of the nodes in the graph: it is closely related to the random walk, so they were also important when defining the diffusion operator. In general terms, when we want to perform a MRA over a graph, the neighbourhood relationship may be a nice starting point.

3.2.4. Reshaping our matrices

In order to study TMs with Matlab we have to reshape them. To begin with, we could reshape it as a vector and combine different vectors to build another matrix. Thus we could study closely some kind of temporal correlations. We used *New_tm1=reshape(TM,1,144);*

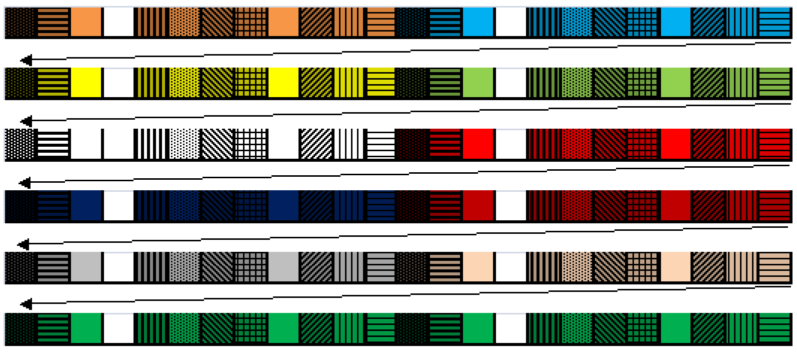


Fig. 3.6 With the reshaping we get a 1x144 vector. Each cell represents a different in-out combination.

Now we can combine different TMs by joining them as row vectors. To make this, we used the function *combine*. We vectorized the twelve matrices for an hour, and then combined them: *new_matrix= combine(new_tm1,new_tm2, new_tm3, [...] new_tm11);*

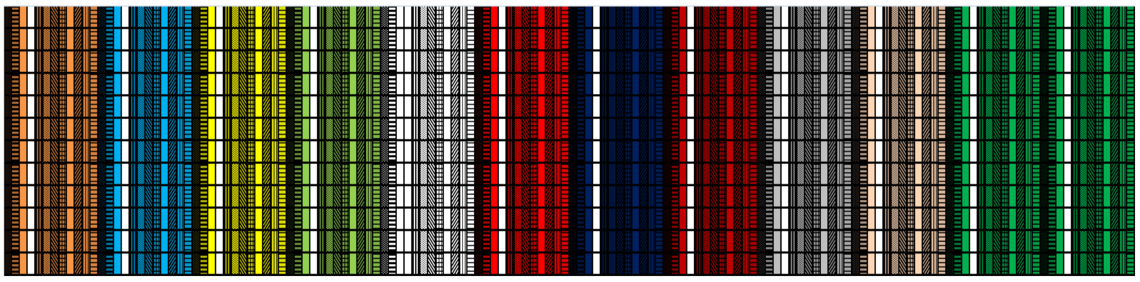


Fig. 3.7 New sample of combined vectorized matrices.

In Fig 3.7 we have an original sample in each row (vectorized, of course) (code in Annex A.XII). So, in each row we have the 144 possible combinations of origin-destination pairs. In order to calculate the temporal correlation, and even though we have a matrix instead of a vector, each column represented the values for traffic in twelve consecutive matrices for each ingress-egress pair and in a whole hour. So, from this structure, we only had to compute the *autocorr* taking the columns.

In addition, Matlab has another correlation function (*xcorr*) which lets us use matrices instead of vectors. However, the computational cost is higher, because it is able to calculate the correlation between the elements of a whole matrix, not only with a vector as *autocorr* does.

3.3. Time correlation

The second step was to start looking for a new diffusion operator, in order to take profit of the time correlations between the traffic matrices series. So, we start from the hypothesis that there may be some time correlation between the amount of traffic in a node at time *t*=*n* and *t*=*n*+*1*, i.e. in consecutive samples. Following this line of research we will build an operator based on this autocorrelation and we will see the diffusion and the compressibility. We will also compare the results with the obtained in chapter 2.

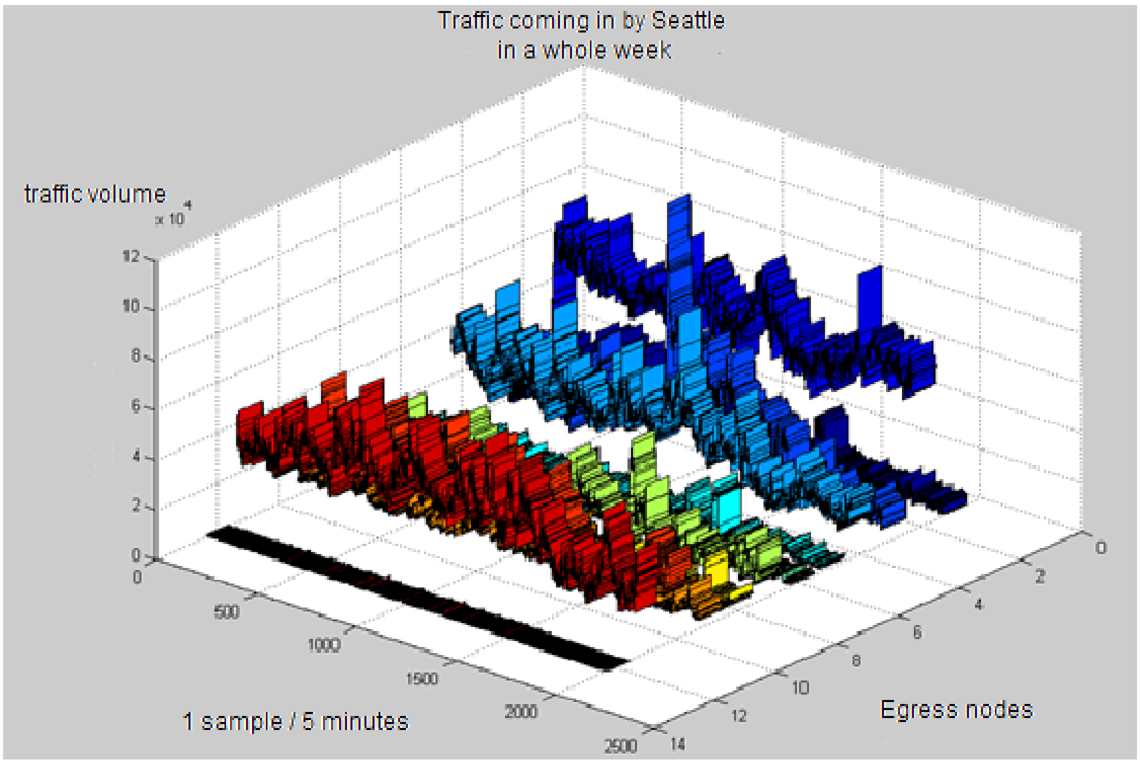
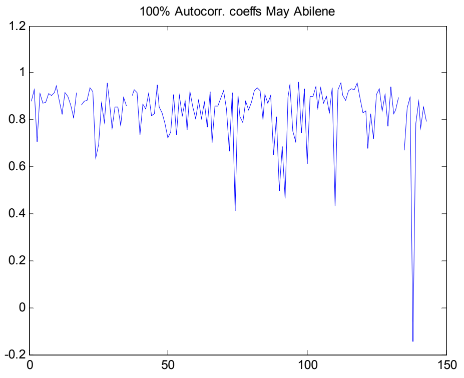


Fig. 3.8 Volume of traffic coming in the network from Seattle in a whole week for Abilene, June 2004. Note that behaviours are different, but in each ribbon we can infer some regularity.

Before searching for any interesting autocorrelation value we can see the daily behaviour or this new set of data. First of all, let's visualize each pair nodes time series. This may be useful to intuitively understand the correlation values obtained later.

The regularities in these ribbons indicate high autocorrelation values, while irregularities like the one in the second ribbon (electric blue) can imply a lower correlation value. And what is more, as we are visualizing a whole week, we can differentiate the day-night cycles and we can notice higher traffic volumes working in days (from Monday to Friday) than in weekends, as expected. We will now describe a pre-processing of the data.

When we had calculated the correlation values for each pair we join them together in a vector. To conclude its autocorrelation study we drew a threshold, discarding the outliers X-AtM5, AtM5-X, IPLS_SNVA, ATLA_ATLA, and NYCM_SNVA pairs. They were the pairs which behaviour was further than the general lines.



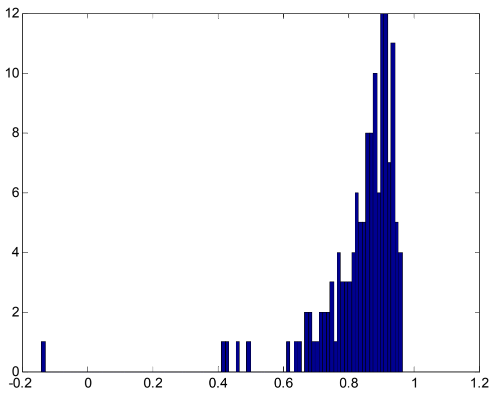


Fig. 3.9 Remaining 144 mean coefficients for May.

In the figure 3.9 we have built a vector with the mean of each in-out pair correlation except those already specified. In numbers, we have passed from 144 mean coefficients to 117 (we have discarded approximately the 10% of coefficients as outliers). From this new "optimized" vector, we find a Mean=0.8338 and a Standard Deviation=0.1317. We consider that this deviation tells us that the mean is reliable.

In the left side of the figure 3.9 we see the vector which contains the autocorrelation means for a month. However, the low peak at the end and maybe a pair or three more points are decreasing the mean. That is why we look at the histogram (right). We can see some sort of discontinuity before 0,6; and for that reason we will discard more outliers. Searching lowest values and discarding them (6,9444% of the coefficients) we obtain a new Mean= 0,8537 and a new Standard Deviation= 0,0770. Not much the mean but the deviation increases a lot the reliability of the estimation.

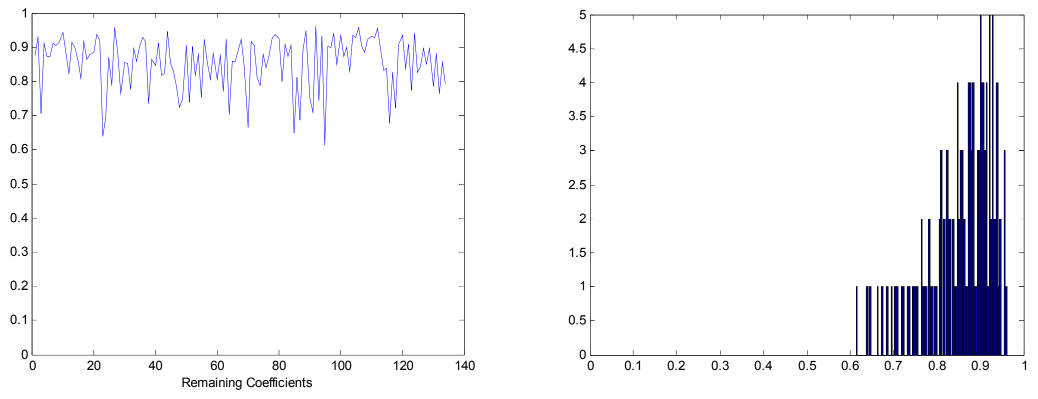


Fig. 3.10 New remaining mean coefficients for May.

As we can see in figure 3.10, discarding the worst samples we get a minimum mean of approximately 0.56. Nevertheless, this “mean of means” is quite good. We get an actual mean of approximately 0.86. This value is quite good (it shows a high similarity between samples) and will be included in our operator as the temporal correlation between samples.

Now we search this mean value for GÉANT. The main problem with GÉANT is the continuity of its samples in time. One of the longest continuous sets of data covers 20 days in Febuary 2005. We can see these means vector in figure 3.11.

The main difficulty we found in GÉANT’s means vector was the less homogeneous distribution of the in-out pairs means. Besides, as we had less samples (4 per hour instead of 12 and 20 days instead of 30) it did not seem a good idea to discard any value, so we can keep the significancy of the results. The image shows some low peaks which reaches autocorrelations under 0,5. However, the many high values will smooth this *a priori* bad mean. In conclusion, we found a Mean= 0,6624 and a Standard Deviation=0.1683.

This time the deviation decreases the reliability of the results, in relation with Abilene. Moreover, the temporal correlation between our series is not as stable as Abilene’s, too. To sum up, we can say that the variability on GÉANT data is higher, so its correlation is lower. This may be because it is a bigger network (more nodes) which links subnetworks (the National academic networks), as a backbone network. This makes that the patterns of traffic show more changes of behaviour. As in Abilene’s case, and for further researches, we will use this found value in the next point.

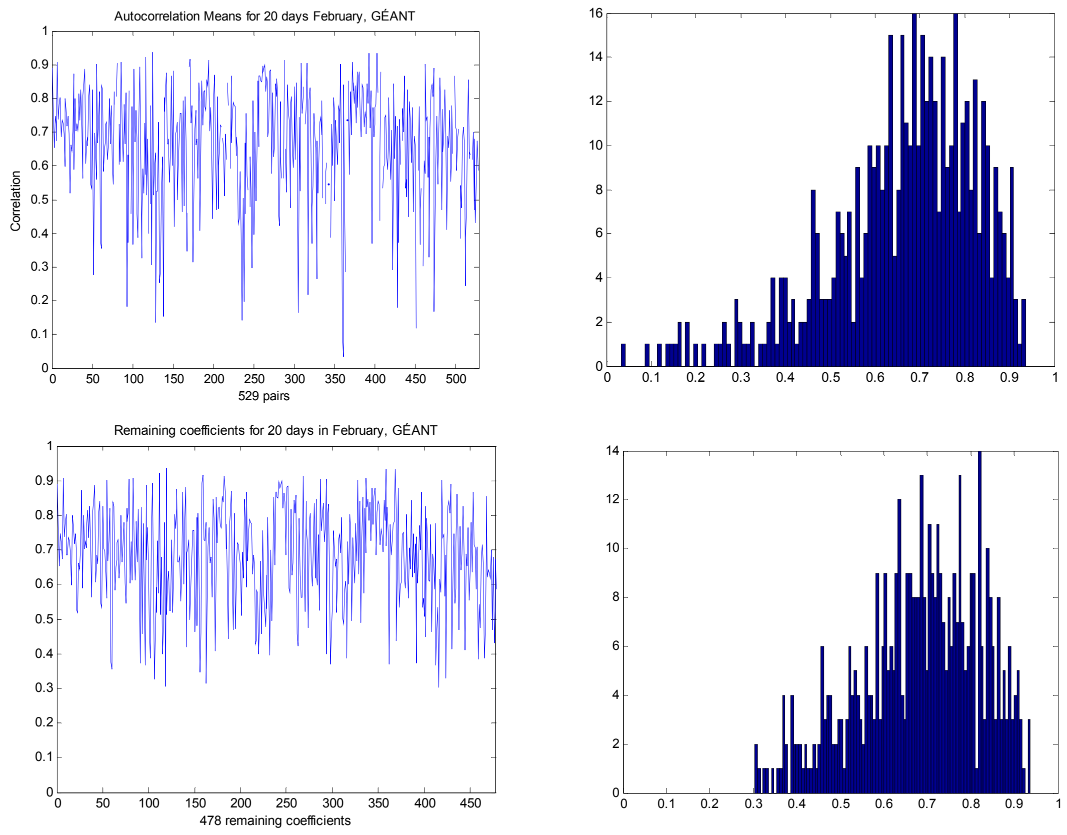


Fig. 3.11 Up: Remaining mean vector of GÉANT. In the histogram we set the threshold at 0,3 (discarding 9,4518% of the values).
Down: the result of this compression. The new Mean= 0,6826 and the new Standard Deviation= 0,1401.

3.4. Looking for a new diffusion operator

We have now to define our time-correlation-based diffusion operator. Once defined, the normal procedure is to develop the diffusion tree, and set the epsilon (precision) in order to perform the MRA. It is important to check if we can reconstruct the original sample from the coefficients obtained from the diffusion and display them as images. After that, calculate the difference or error due to the compression (we call compression to the discarding of eigenvalues). For more information about building diffusion wavelet trees, see Annex A.III.

In chapter 2 we have seen the results of using a topology-based or a gravity-based operator. Now we want to include the similarities between samples in our operator, so we will use the time correlation results for this purpose. The original idea was to build up a new sort of 3D operator. We can imagine the samples as slices: just us movie photograms, which are different at each time instant, but give the impression to be continuous when watched sequentially. Our operator is going to be composed by slices, and there must be a parameter to relate the similarity between slices. Each slice would represent a traffic matrix, and we would set the correlation (similarity) between each nodes pair as the link between one sample in t=0 and the next one in t=1.

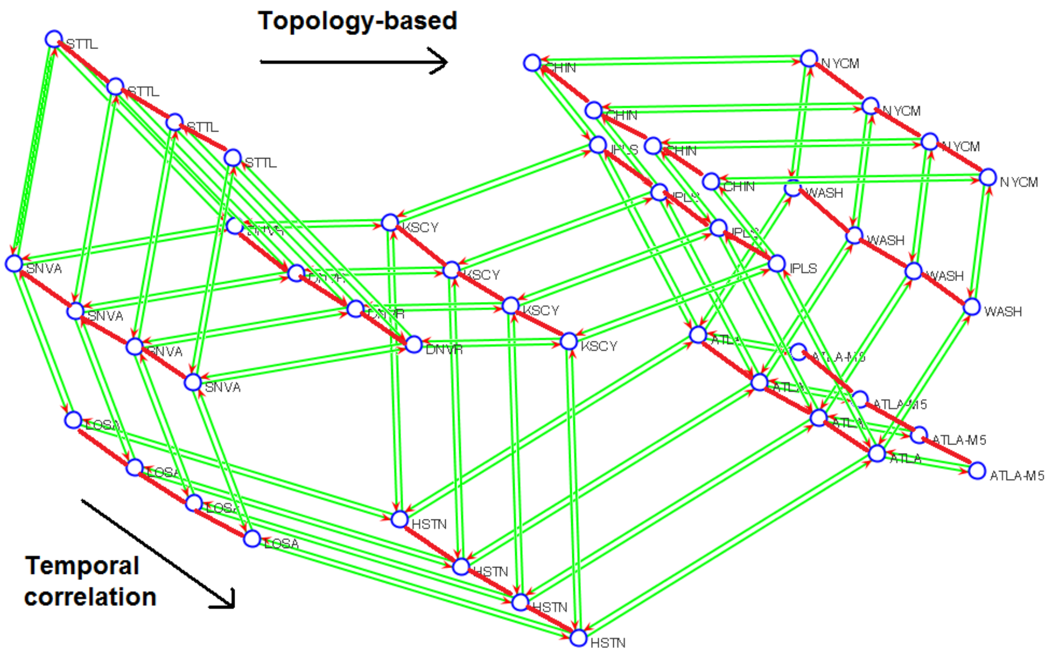


Fig. 3.12 Graph of our new 3D operator which relates both topology (space) and time correlation (similarities in time)

Imagine that there are four slices of the original graph, but each one corresponds to a different time sample. We add the links between a node and itself-in-next-sample and will give that links the value of the mean of the correlations we have found before. Its complexity grows when we try to translate this into Matlab. If we see it like a graph it may be difficult. We tried to represent as a matrix, since adjacency matrices give us information of the topology.

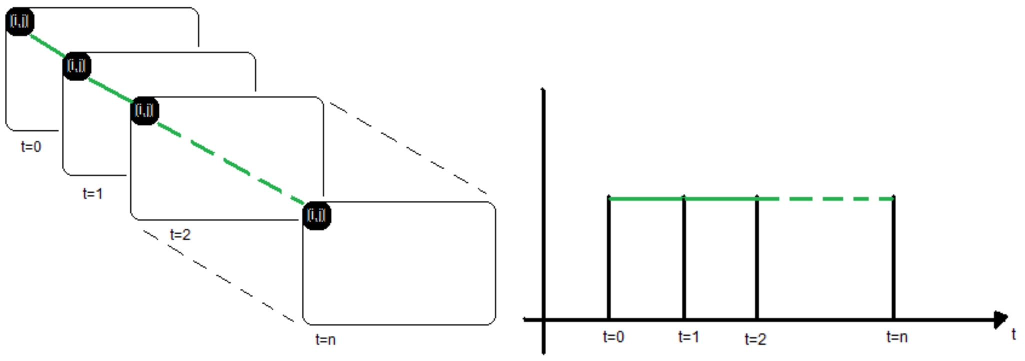


Fig. 3.13 The idea of our 3D graphic translated into a 3D matrix. Green links stand for correlation. On the right, see the TMs like slices in time.

There are some ways of building 3D matrices in Matlab, but the main problem will come when trying to develop the diffusion wavelet tree. However, the solution is quite simple. Let's imagine a plain graph (a graph whose edges does not cross) and planish the subsequent-in-time graph. Then, each node with itself would be adjacent, so we reshape this 3D graph into a 2D plain and very big graph.

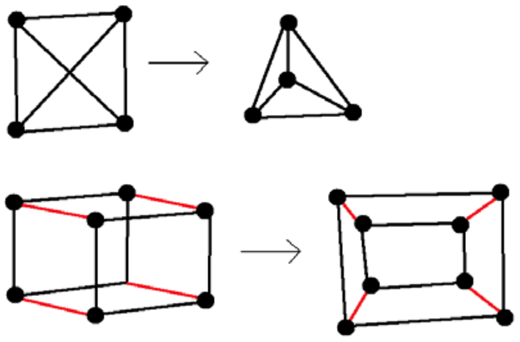


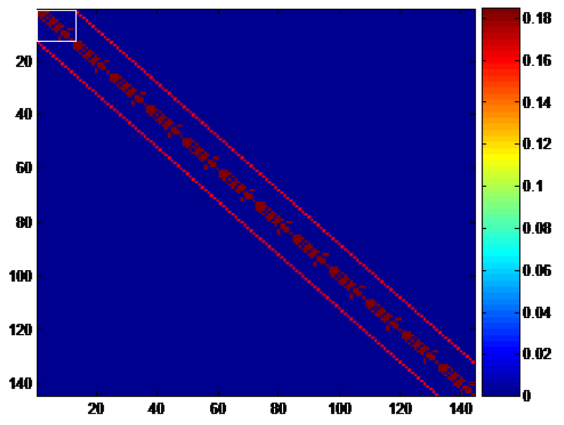
Fig. 3.14 How to make a plain graph. The "cube" can be interpreted as two slices and its nodes linked to each other between slices, just as we want our operator. This operation can be done as many times as we want.

Starting from this new interpretation of our 3D graph, we were able to build the new operator. For this purpose we recovered the adjacency matrix of our graph, where A[i,j]=1 if i and j are neighbours. Besides, we already thought to take into account the cases where i=j (the traffic flowing from the node to the node itself or *autotraffic*). Remember that from each node hang other nodes or even whole networks, so there will be special flows that increase the load of the node but does not travel along the main network.

To that adjacency matrix, we have to add the link which is tying the nodes with their future state nodes. So, if we are considering new samples of an hour, the operator will be formed by a diagonal of twelve adjacency matrices, with two additional diagonals for autocorrelations and a main diagonal for autotraffic. Once we have the operator, we must normalize it over its highest eigenvalue, so that when we compute the coefficients, the highest eigenvalue is one.

Note that the colourbar ranges from 0 to 0,18 approximately (in figure 3.15). That is because the operator has been normalized by its highest eigenvalue. To build the operator some code has been developed. It can be found in annex A.IV.

Once we have the operator we must compute the diffusion wavelet tree. The explanation would be given over the sample tree in the next point (more tests with our new time-correlation-based diffusion operator in A.XXI, A.XXII, A.XXIII).



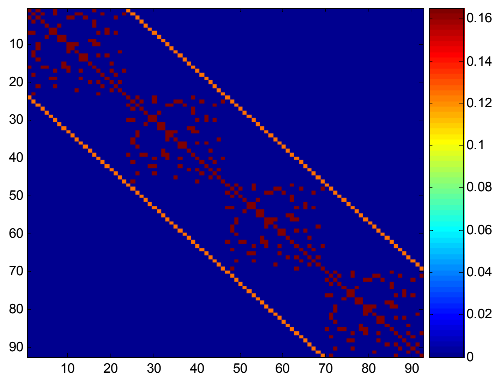


Fig. 3.15 Our two-redimensioned 3D diffusion operator. A central diagonal of adjacency matrix with autotraffic (repetitions of the white square in left) and two additional linking each node with itself in the next sample. Left, Abilene's; right, GÉANT's. Notice than in GÉANT's there are four repetitions (1 hour). Abilene¡s operator is bigger, so we expand the image to better seeing its structure.

3.5. The new operator's Diffusion Wavelet Tree

Let's see the diffusion process with our new temporal operator. The way it works and the interpretations will be explained through an example (Abilene, June 21st 2004, from 12:00 to 12:55).

First of all we load the Abilene adjacency matrix. Then we build the operator: *temporal_matrix_abilene* needs the matrix to repeat on the diagonal, the number of samples we will use and the mean correlation (see section 3.3 Time autocorrelation). Then we normalize by its higher eigenvalue. And the next step is computing the tree.

The first thing to take into account is that we can compute a partial tree or a full tree. The difference is that in partial decomposition we are working with the lowest frequency signal (see figure 2.3). The full decomposition computes the approximation and details of each decomposed signal, i.e. the first high frequency signal (first details) will be dissected into its details and approximations, too. This procedure is similar to the Wavelet Packet Transform [16,26].

The second thing is that we have to give the function a delta basis (combination of ones) to be able to compute the diffusion. This basis is given by *speye(144)*, a diagonal of 144 ones. The next is to specify our diffusion operator, the level of decomposition we want to obtain and the epsilon (precision) of this decomposition.

Table 3.1 Code and Tree for computing the diffusion wavelet tree. In our case we will perform the standard Partial Diffusion Wavelet.

```
load abileneAdjacency;
diffusion=temporal_matrix_abilene(abileneAdjacency,12,0.87);
diffusion2=diffusion/max(eig(diffusion));
Tree = DiffusionWaveletTree( 'Partial', [], speye(144), diffusion2, 15, 1e-15);
V_0:        Index: {1}(1)    Level: 0  # of Fcns: 144   Freq Range: [0 1]
V_1:        Index: {2}(1)    Level: 1  # of Fcns: 144   Freq Range: [1e-015 1]
W_1:        Index: {2}(2)    Level: 1  # of Fcns: 0     Freq Range: [0 1e-015]
V_2:        Index: {3}(1)    Level: 2  # of Fcns: 144   Freq Range: [3.16228e-008 1]
W_2:        Index: {3}(2)    Level: 2  # of Fcns: 0     Freq Range: [1e-015 3.16228e-008]
V_3:        Index: {4}(1)    Level: 3  # of Fcns: 139   Freq Range: [0.000177828 1]
W_3:        Index: {4}(2)    Level: 3  # of Fcns: 5     Freq Range: [3.16228e-008 0.000177828]
V_4:        Index: {5}(1)    Level: 4  # of Fcns: 115   Freq Range: [0.0133352 1]
W_4:        Index: {5}(2)    Level: 4  # of Fcns: 24    Freq Range: [0.000177828 0.0133352]
V_5:        Index: {6}(1)    Level: 5  # of Fcns: 63    Freq Range: [0.115478 1]
W_5:        Index: {6}(2)    Level: 5  # of Fcns: 52    Freq Range: [0.0133352 0.115478]
V_6:        Index: {7}(1)    Level: 6  # of Fcns: 23    Freq Range: [0.339821 1]
W_6:        Index: {7}(2)    Level: 6  # of Fcns: 40    Freq Range: [0.115478 0.339821]
V_7:        Index: {8}(1)    Level: 7  # of Fcns: 9     Freq Range: [0.582942 1]
W_7:        Index: {8}(2)    Level: 7  # of Fcns: 14    Freq Range: [0.339821 0.582942]
V_8:        Index: {9}(1)    Level: 8  # of Fcns: 4     Freq Range: [0.763506 1]
W_8:        Index: {9}(2)    Level: 8  # of Fcns: 5     Freq Range: [0.582942 0.763506]
V_9:        Index: {10}(1)   Level: 9  # of Fcns: 2     Freq Range: [0.873788 1]
W_9:        Index: {10}(2)   Level: 9  # of Fcns: 2     Freq Range: [0.763506 0.873788]
V_10:       Index: {11}(1)   Level: 10 # of Fcns: 2     Freq Range: [0.934766 1]
W_10:       Index: {11}(2)   Level: 10 # of Fcns: 0     Freq Range: [0.873788 0.934766]
V_11:       Index: {12}(1)   Level: 11 # of Fcns: 1     Freq Range: [0.966833 1]
W_11:       Index: {12}(2)   Level: 11 # of Fcns: 1     Freq Range: [0.934766 0.966833]
V_12:       Index: {13}(1)   Level: 12 # of Fcns: 1     Freq Range: [0.983277 1]
W_12:       Index: {13}(2)   Level: 12 # of Fcns: 0     Freq Range: [0.966833 0.983277]
V_13:       Index: {14}(1)   Level: 13 # of Fcns: 1     Freq Range: [0.991603 1]
W_13:       Index: {14}(2)   Level: 13 # of Fcns: 0     Freq Range: [0.983277 0.991603]
V_14:       Index: {15}(1)   Level: 14 # of Fcns: 1     Freq Range: [0.995793 1]
W_14:       Index: {15}(2)   Level: 14 # of Fcns: 0     Freq Range: [0.991603 0.995793]
```

In the decomposition it is important to understand that Vs are approximations and Ws are details. It is also important to see how the frequency range of the decomposition tends to the whole. But, what is more, we will analyze the number of Functions (i.e., the number of eigenvalues/eigenvectors for each subspace). Here is where we will find the value to set our threshold (ε) and the subsequent compression and error. In this example the decomposition can stop at the 11th scale, since V_11 is composed by just one single eigenvalue (λ=1) and therefore there is no point in going further.

Many decompositions have been made to analyze the effect of this precision parameter, e.g. with an $\varepsilon=10^{-5}$ the tree gives us a different decomposition (more tests in Annex A.XX). The ideal epsilon would be that which gathers a single eigenvalue in a subspace. However, we can play with this parameter to have an idea of the distribution of the eigenvalues. Nevertheless, in our experiments we order the eigenvalues from the highest down and this separation of eigenvalues in subbands looses importance, as the results are reliable, too.

Now we can visualize the diffusion we have achieved with our new operator. In figure 3.16 we can see how the decomposition is made and how the fact of having less information in each step produces this blurring effect. Finally, V_j has just one eigenvalue. What we get is the mean of the TM (like the zero-frequency coefficient in Fourier). For the details W_j (with 0 eigenvalues) we get a zero matrix.

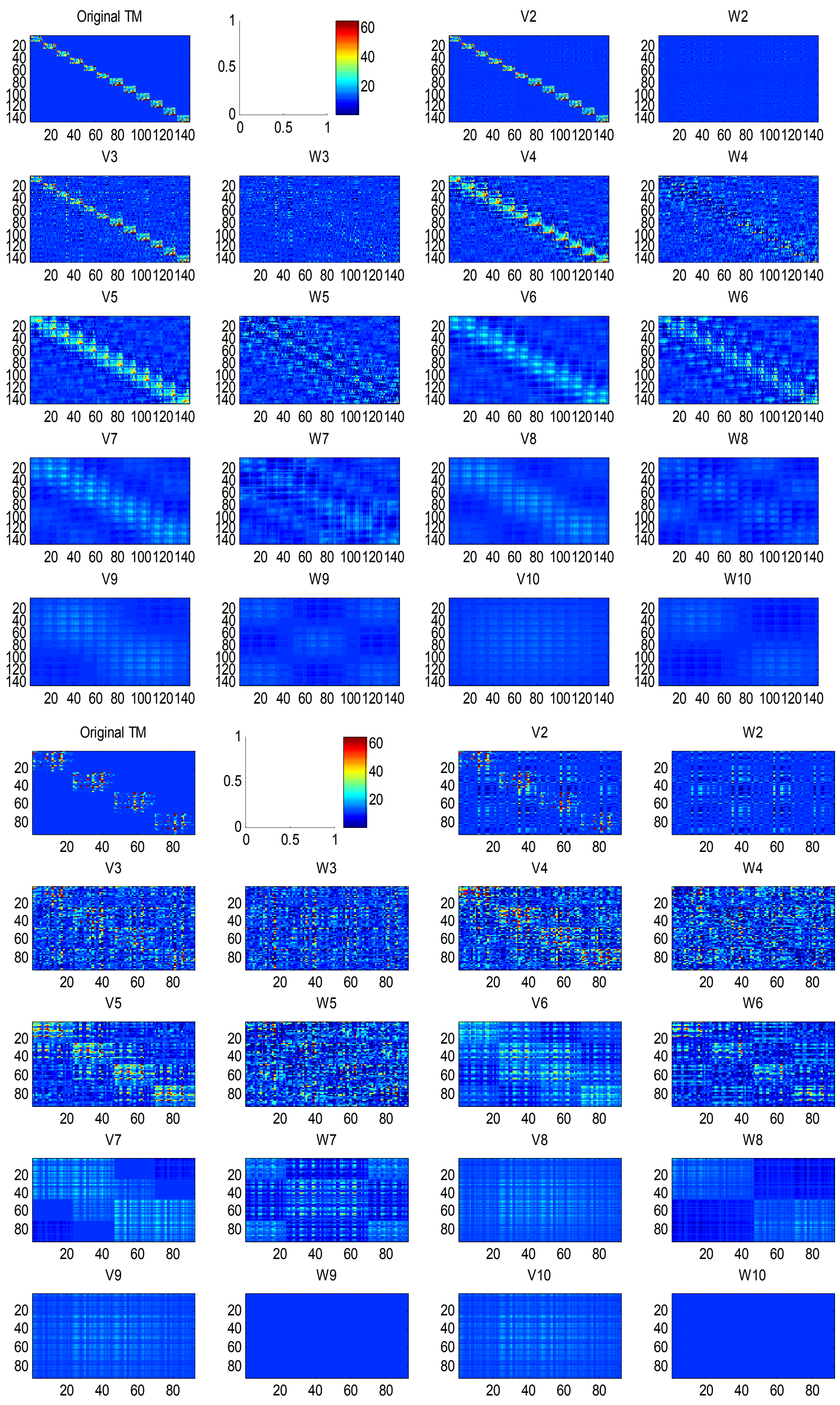


Fig. 3.16 Diffusion wavelet tree visualization for approximations and details.

Figure 3.16 shows the diffusion with Abilene (up, 12 matrices in diagonal, from 12:00 to 12:55, June 21st 2004) and GÉANT (down, 4 matrices in diagonal from 20:00 to 20:45, February 22nd 2005).

A good way to see the diffusion is to reconstruct the TMs, repeating the decomposition process in some sort of an inverse way: from the coefficients, we will get the details and approximations that compose a certain level of approximation to the original TM.

3.6. Compressibility and error

After having achieved a satisfactory decomposition, we will perform an analysis of the compressibility of the input TM.

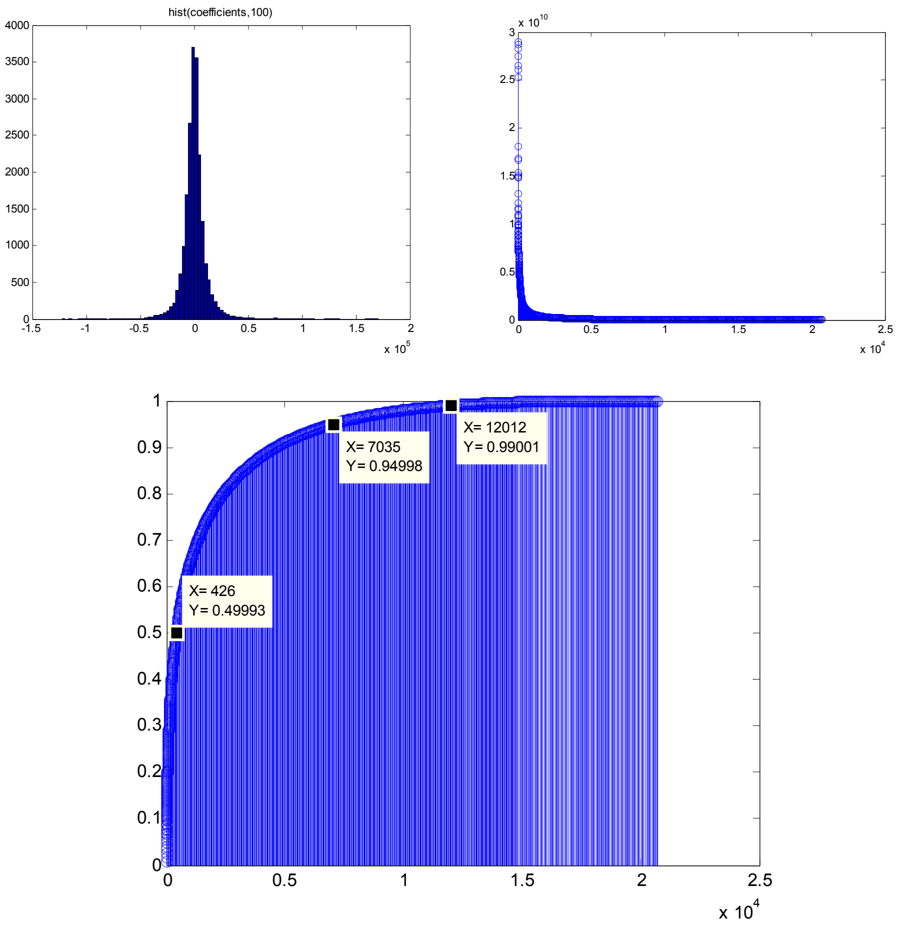


Fig. 3.17 Distributions for the coefficients and the energy. Upper left: the histogram for the coefficients without ordering them. We note that the distribution is Gaussian-alike. Next (upper right) we have the representation of all the ordered coefficients (horizontal axis). We see that a few coefficients gather the most of the energy. Finally (down) we have the cumulative representation the energy in terms of the ordered coefficients (horizontal axis).

In the figure of the left we can see a quite gaussian distribution of the coefficients. In the right, we have plotted the histogram of the square of the coefficients. It can be read as it is the energy concentration (power). It is quite good to have the 99% of the energy with a little more than the half of the coefficients. In addition, a high compression can be achieved. Moreover, if we think that 95% of the energy is still a good percent, we only have to maintain 1/3 of the coefficients.

Table 3.2. Comparison of the percentage of coefficients which gather different amounts of energy for different epsilons in Abilene (May 2004) and GÉANT (February 2005).

% of energy	***Abilene ε=10^{-15}***	***Abilene ε=10^{-5}***	***GÉANT ε=10^{-15}***	***GÉANT ε=10^{-5}***
50%	2.1%	3.74%	3.06%	5.15%
90%	33.93%	42.1%	31.45%	31.43%
99%	57.93%	64.29%	53.13%	64.19%

In this table we can see some tendency between all the variables. In addition, we see that with a higher epsilon the compressibility decreases (for these specific traces). Some differences between Abilene and GÉANT are noticeable, but although the differences in topology and sampling, the results are quite similar. However we must take into account that the number of samples in GÉANT series was lower, too. In the figure 3.21 we can see the dispersion (similarities) of these four variables:

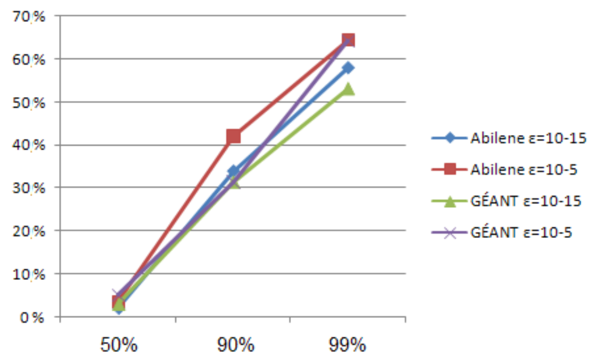


Fig 3.18 Similarities (dispersion) of the amounts of energy related to the percentage of coefficients taking into account different networks and epsilons.

While computing the error it must be said that we will use the *Mean Squared Error* (*MSE*). This error is a way to quantify the difference between the original and the reconstructed coefficients, and can be understood as the second moment of the error. To see the code used to the MSE, see Annexes A.V and A. XXIV.

Analogously to the compression estimation, we will normalize the MSE (code for MSE computing in Annex A.XXIV) in terms of power (figure 3.19).

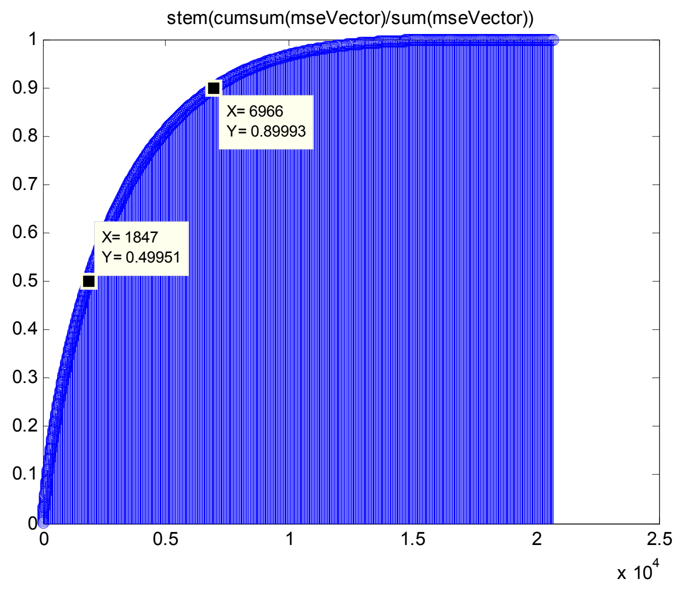


Fig. 3.19 MSE distribution in terms of energy. (Abilene May 2004, ε=10^{-15}).

Table 3.3. Comparison of the percentage of coefficients with the MSE for different epsilons in Abilene (May 2004) and GÉANT (February 2005).

% of MSE energy	*Abilene ε=10^{-15}*	*Abilene ε=10^{-5}*	*GÉANT ε=10^{-15}*	*GÉANT ε=10^{-5}*
50%	8,91%	10,7%	7,8%	10,4%
90%	33,6%	36,3%	29,96%	35,39%

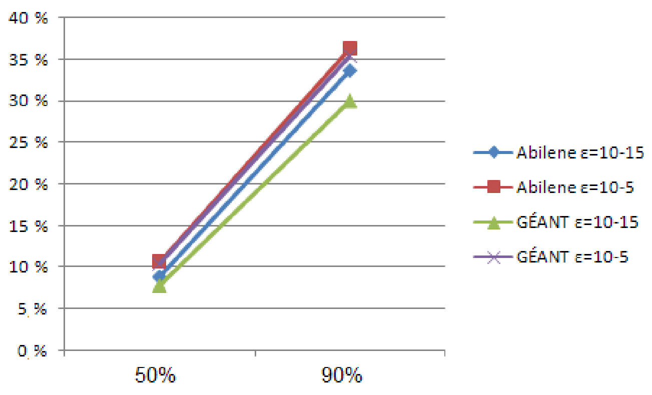


Fig 3.20 Similarities (dispersion) of the energy induced by the MSE related to the percentage of coefficients taking into account different networks and epsilons.

We also see some trends in this case. Although the datasets are different, the results of the MSE induced by the compressibility are similar. This is an interesting indicator because GÉANT and Abilene have different topologies, time-sampling, etc. It seems that there are some similarities in the results obtained with different operators and with different time series. We have the intuition of that the lower the epsilon is, the higher the error may be.

To conclude our study we have wanted to compare the MSE of different time series. The original idea was to compute the error for an hour series, compare this error with the error for a day series, and then for a week or even a month series, in order to confirm that the length of the time series affects minimally to the compressibility and the error. However, at time of computing MSE a lot of CPU power is needed, so we have only been able to compute the MSE for a single day (in our case we used a PC with Intel Core 2 Quad processor working at 2.5 GHz, with 4 GB of RAM and the computing time has approximately been of nearly 2 hours for a single hour in Abilene, and 15 minutes for an hour series in GÉANT). However, we have analyzed different days for the two datasets to compare them.

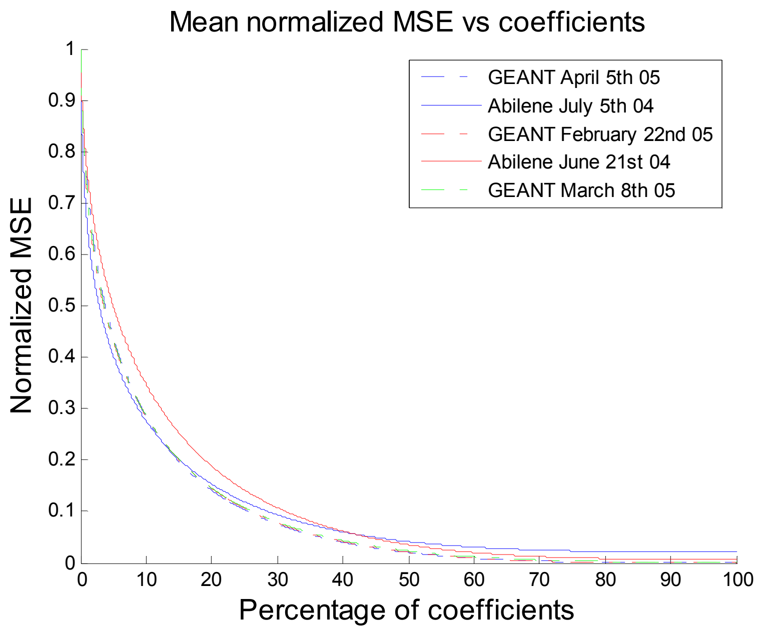


Fig 3.21 Mean normalized MSE for different days in GÉANT and Abilene

In figure 3.21 we can see, in dotted-lines, three days for GÉANT and in continuous lines two days for Abilene. The distributions for GÉANT are quite similar, and for Abilene vary a little more. In addition, at the 100% of the coefficients, Abilene's MSE is higher. This may be because the sampling time-slot is shorter than GÉANT`s. Another interesting conclusion is that the curve decreases rapidly: with the 30% of the coefficients (at GÉANT) we are achieving only a 10% of error.

If we compare these results with the obtained with the gravity and topology operator (figure 2.14) we see that the MSE of our operator is higher (for example, with gravity operator we have less than 10% of error with just 10% of the coefficients) and therefore it does not compress as much as the old operators. Nevertheless, we have to say that the trace of Figure 2.14 stands for a whole month, while Figure 3.21 shows results of analysis of just a single day. However, we have some intuition that the results for a month (when computing them, and taking into account the cost it will have) will return similar results.

Finally, although we are not sure that the dates are actually certain (due to anonymization) we can state that they are not weekend days and they are the same day in the week. We have obtained one more day for GÉANT because the computing for Abilene needs more CPU power.

In conclusion, the time-correlation-based operator seems not to give us better results than the ones we obtained with previous operators. In addition, to compute the processes a lot of CPU power is required, so we can affirm that this is not the better option. However, it has been interesting to include the time in our operator. Maybe mixing gravity model with the temporarily of series we would get better results.

CONCLUSIONS

To conclude, we first want to highlight the importance of TMs in traffic engineering, since they relate the nodes of a network with the traffic flowing through them. We have also presented the application of TMs in anomaly detection, routing and load balacing, etc. However we face two problems in order to get these matrices. First, there is the lack of public data. The second matter is how to measure real matrices. For this purpose we can use two methods: SNMP or Netflow (or NetFlow-like applications). With Netflow we have accurate data (ingress node, egress node, specific volumes of traffic, etc) at the expense of complexity and router CPU power; with SNMP we have more general data (aggregated flows in each node, etc) but is easier to measure. In addition, SNMP is present in every router and SNMP measures are easier to perform, while netFlow is more complicated to configure.

Our long term goal is to get a model which allow us to predict traffic behaviour. We have also presented the importance of models: in this field we must highlight the tomographic comparisons and gravity model as a start point, and the inferring process used when only SNMP measurements are available (an inference procedure can be followed in order to get an estimation of the TM from the incomplete SNMP data).

On the other hand, mathematical tools have opened the researchers new paths for study. We have described a multiresolution analysis of traffic matrices and models. This multiresolution allows us to analyze and visualize the signals at different scails of detail. This scaling allow us to rebuild the TM, just as if we wre zooming in and out in a way that is adapted to the underlying graph structure. To achieve this multiresolution we have presented wavelets, and their adaptations to be used in two dimensions (varying their classical use to be developed over images, e.g. JPEG 2000). But we cannot understand a graph like an image, because the relationship between the pixels of an image is very different from the neighbourhood of the nodes in a network. This difference bind us to use anoher version of wavelets in order to develop the multiresolution analysis over the network: diffusion wavelets. Moreover, we have described the DW in 2D and used it in analysis of TMs.

The multiresolution will give us the original sample splitted in details (high frequencies) and approximations (low frequencies). Implementing the multiresolution with a filter bank, the output of the fist low pass filter will be the input of the next scale. After the diffusion process (which will give us the coefficients of the original sample) we can compute the compression deciding how many detail coefficients and approximations we can discard. Then we have also to check the error obtained after this compression. All these previous results have been described in a paper presented in JITEL 2009 (Jornadas de Ingeniería Telemática) held in Cartagena (Spain) in September 2009 [10].

Then we presented our experiments and the software we have used for this purpose. Although we have not used TOTEM too much, we can say that it is a powerfol tool to traffic engineering, since it is capable to simulate a wide range of scenarios. On the other hand, we present some Matlab features, as it is the software we have used to develop all our tests. Then we present the process of reshaping and combining matrices as a way to build new datasets which allows us to investigate its temporal evolution by their column submatrices.

Finally, we described how to build a 3D diffusion operator that we thought could capture the time correlation present in the TM datasets. We have structured the graph as slices for each temporal sample, where each node was linked to the same node an instant later by some temporal correlation. We have seen that the correlation between the samples is quite stable, so we have looked for the mean of the correlations to find a specific value for this graph edges. We have developed the diffusion process with our new operator and we have compared the results with the obtained with other operators.

We have obtained similar results for Abilene and GÉANT. This, and although the results brought by our new operator are not as good as we expected, gives us some confidence on the validity of the approach, since previous works also found the same feature. It also gives us the confidence to believe that a general model (when found) could be use for both academic networks.

There is still a lot of work to be done: transforming the compressibility of traffic matrices in a physical model; using this model to solve inference, synthesis, prediction and anomaly detection problems; looking for the relationship of diffusion wavelets sparse model with others like gravity model; looking for new diffusion operators (specially the ones which have some relation with the routing matrices); researching about the representations of different topologies through MRA; and combining the Gravity Model with temporal correlations.

Finally, regarding the ambientalisation of our work, we can confirm that the impact of our research over the environment is positive. As we are looking for tools for improving the performance of networks, we can assume that there will be an assocated saving of energy [14] due to this optimization for example, by performing power-consumption-aware routing. However, the computational cost for this processes (specially if we are working with long-time sets) is high, but we can state that the synthesis of data (as well as the sparsity of this data) may save little amounts of energy.

BIBLIOGRAPHY

[1] Roughan, M. Thorup, M. and Zhang, Y., “Traffic Engineering with Estimated Traffic Matrices”, *Proceedings of the USENIX/ACM Internet Measurement Conference (Miami, Florida*). pp. 248-258. (2003).

[2] Leprope, J., Balon, S. and Leduc, G., “TOTEM: a TOolbox for Traffic Engineering Methods”, Poster and Demo Session of *INFOCOM’06 (Barcelona, Spain)*. (2006).

[3]Balon, S., Lepropre, J., Delcourt, O., Skivée, F. and Leduc, G., “Traffic Engineering an Operational Network with the TOTEM Toolbox”, *IEEE Transactions on Network and Service Management*. 4 (1). pp. 51-61. (2007).

[4] Coates, M., Pointurier, Y. and Rabbat, M., “Compressed network monitoring for IP and all-optical networks”, *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement (San Diego, California)*. pp. 241-252. (2007).

[5] Crovella, M. and Kolaczyk, E., “Graph Wavelets for Spatial Traffic Analysis”, *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*. 3. pp. 1848-1857. (2003).

[6] Rincón, D., Roughan, M., and Willinger, W., “Towards a Meaningful MRA of Traffic Matrices”, *Internet Measurement conference. Proceedings of the 8th ACM SIGCOMM conference on Internet Measurements (Vouliagmeni, Greece)*. pp. 331-336. (2008).

[7] Wild, M., “Nonlinear Approximation of Spatiotemporal Data Using Diffusion Wavelets”, in *Computer Analysis of Images and patterns*. Springer Berlin/Heidelberg. 4673. pp. 886-894. (2007).

[8] García de Jalón, J., Rodríguez, J.I. and Vidal, J., “Aprenda Matlab 7.0 como si estuviera en primero”, *Tutoriales UPM*. (2006). http://meteo.fisica.edu.uy/?download=matlab70primero.pdf

[9] Coifman, R.R. and Maggioni, "Diffusion Wavelets”, in *Applied and Computational Harmonic Analysis*. 21 (1). pp. 53-94. (2006).

[10] Rincón, D., Torres, J., “Contribuciones al análisis multirresolución de matrices de tráfico”, *VII Jornadas de Ingeniería Telemática JITEL 2009 (Cartagena, Spain).* pp. 413-420. *(2009).*

[11] Lakhina, A., Papagiannaki, K., Crovella, M., Diot, C., Kolaczyk, E. and Taft, N., “Structural Analysis of Network traffic Flows”, *Joint International Conference on Measurement and Modeling of Computer Systems (New York, USA)*. pp. 61-76. (2004).

[12] Maggioni, M., “Diffusion analysis of and on graphs and high dimensional data”, Invited talk at *IPAM-UCLA,* (2007). http://www.math.duke.edu/~mauro/Talks/Banff.pdf

[13] Zhang, Y., Roughan, M., Duffield, N., Greenberg, A, “Fast Accurate Computation of Large Scale IP Traffic Matrices from Link Loads”, *ACM SIGMETRICS Performance Evaluation Review (New York, USA)*. 31 (1). pp. 206-217. (2003).

[14] Ceuppens, L., Sardella, A., Kharitonov, D., “Power Saving Strategies and Technologies in Network Equipment Opportunities and Challenges, Risk and Rewards”, *Proceedings of the 2008 International Symposium on Applications and the Internet*. pp. 381-384. (2008).

[15] Uhlig, S., Quoitin, B., Lepropre, J., Balon, S., “Providing public intradomain traffic matrices to the research community”, *ACM SIGCOMM Computer Communication review*. 36 (1). pp 83-86. (2006).

[16] Mallat, S., *A wavelet tour of signal processing*. Academic Press, London 1999.

[17] Fortz, B., Rexford, J., Thorup, M., “Traffic engineering with traditional IP routing protocols”, *IEEE Communication Magazine*. 40 (10). pp. 118-124. (2002)

[18] Nucci, A., Sridharan, A., Taft, N., “The problem of synthetically generating IP traffic matrices: Initial recommendations”, *SIGCOMM Computer Communication Review*. 35 (3). (2005).

[19] F. Chung. “Spectral Graph Theory”, *CBMS Regional Conference Series in Mathematics*. 92. (1997).

[20] Universidad Nacional del Centro de Buenos Aires (UNICEN, Argentina), “Introducción a la transformada Wavelet” http://www.exa.unicen.edu.ar/escuelapav/cursos/wavelets/apunte.pdf

[21] Mathwork’s Matlab Home Site http://www.mathworks.com/products/matlab

[22] TOTEM Project Home Site http://totem.run.montefiore.ulg.ac.be

[23] Roughan, M., “AT&T Labs Tomogravity research” http://research.att.com/viewProject.cfm?prjID=133

[24] Wikipedia, *Spectral Theorem* http://en.wikipedia.org/wiki/Spectral_theorem

[25] Wikipedia, *Discrete Wavelet Transform* http://en.wikipedia.org/wiki/Discrete_wavelet_transform

[26] Wikipedia, *Wavelet Packet Decomposition* http://en.wikipedia.org/wiki/Wavelet_packet_decomposition

[27] Simple Network Management Protocol (SNMP) RFC
http://tools.ietf.org/html/rfc1157

[28] Wikipedia, *NetFlow* http://en.wikipedia.org/wiki/Netflow

[29] Wikipedia, *Multiresolution Analysis*
http://en.wikipedia.org/wiki/Multiresolution_analysis

[30] JPEG 2000 RFC http://tools.ietf.org/html/rfc3745#ref-ISO-JPEG2000-1

[31] Wikipedia, *Point of Presence* http://en.wikipedia.org/wiki/Point_of_presence

[32] Abilene Network http://www.internet2.edu/network/

[33] GÉANT Network http://www.geant.net/

GLOSSARY

2D DW: Diffusion wavelets in two dimensions

DoS: Deny of Service

DW: Diffusion Wavelet

DWT: Discrete Wavelet Transform

GM: Gravity Model

GUI: Graphic User Interface

ISP: Internet Service Provider

MAN/WAN: Metropolitan / Wide Area Network

MIB: Management Information Bases

MRA: MultiResolution Analysis

MSE: Mean Square Error

PoP: Point of Presence

QoS: Quality of Service

SNMP: Simple Network Management Protocol

TM: Traffic Matrix

TOTEM: Toolbox for Traffic Engineering Methods