2011

B-splines in EMD and Graph Theory in Pattern Recognition

Qin Wu
West Virginia University

B-splines in EMD
and
Graph Theory in Pattern Recognition

Qin Wu

Dissertation submitted to the
Eberly College of Arts and Sciences
at West Virginia University
in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy
in
Mathematics

Sherman D. Riemenschneider, Ph.D., Co-chair
Cun-Quan Zhang, Ph.D., Co-chair
Eddie Fuller, Ph.D.
Mary Ann Clarke, Ph.D.
Arun A. Ross, Ph.D.

Department of Mathematics

Morgantown, West Virginia
2011

Keywords: B-spline, EMD, pattern recognition, weighted graph, centrality, clustering, postman tour, route-pair cost

ABSTRACT

B-splines in EMD and Graph Theory in Pattern Recognition

Qin Wu

With the development of science and technology, a large amount of data is waiting for further scientific exploration. We can always build up some good mathematical models based on the given data to analyze and solve the real life problems. In this work, we propose three types of mathematical models for different applications.

In chapter 1, we use Bspline based EMD to analysis nonlinear and no-stationary signal data. A new idea about the boundary extension is introduced and applied to the Empirical Mode Decomposition(EMD) algorithm. Instead of the traditional mirror extension on the boundary, we propose a ratio extension on the boundary.

In chapter 2 we propose a weighted directed multigraph for text pattern recognition. We set up a weighted directed multigraph model using the distances between the keywords as the weights of arcs. We then developed a keyword-frequency-distance-based algorithm which not only utilizes the frequency information of keywords but also their ordering information.

In chapter 3, we propose a centrality guided clustering method. Different from traditional methods which choose a center of a cluster randomly, we start clustering from a "LEADER" - a vertex with highest centrality score, and a new "member" is added into an existing community if the new vertex meet some criteria and the new community with the new vertex maintain a certain density.

In chapter 4, we define a new graph optimization problem which is called postman tour with minimum route-pair cost. And we model the DNA sequence assembly problem as the postman tour with minimum route-pair cost problem.

Acknowledgements

First, I would like to express my sincere thanks to my advisor, Dr Sherman D. Riemenschneider and my dissertation co-advisor, Dr Cun-Quan Zhang for their guidance, encouragement, help and support during all these years. It is a great pleasure to work under their supervision. What I learned from them will benefit me all my life.

Second, I'd also like to thank my committee members: Dr Eddie Fuller, Dr Mary Ann Clarke and Dr Arun A. Ross, for their help during the completion of my dissertation.

Finally, I would like to thank the Department of Mathematics and Eberly College of Arts and Sciences at West Virginia University for providing me with an excellent study environment and continual support during my years as a graduate student.

Contents

Chapter 1

Empirical Mode Decomposition

1.1 Introduction

Signal analysis is an important tool in both pure research and practical applications. Traditional data analysis methods such as Fourier analysis, based on the linear stationary assumption have been shown to be efficient for processing of linear and stationary data. However, these methods are less suitable for analyzing nonlinear and non-stationary data. The physically meaningful way to describe the non-linear system is in terms of the instantaneous frequency which reveals the intra-wave frequency modulations[1]. The easiest way to compute the instantaneous frequency of any signal is by using the Hilbert transform. For a real signal $s(t) \in L^p$, the Hilbert transform is defined as

$$\mathcal{H}[s(t)] = \frac{1}{\pi} PV \int_{-\infty}^{\infty} \frac{s(\tau)}{t-\tau}\, d\tau$$

where PV represents the principle value.

With the Hilbert transform, the analytic signal is define as

$$z(t) = s(t) + i\,\mathcal{H}[s(t)] = a(t)e^{i\theta(t)}$$

with

$$a(t) = \sqrt{[s(t)]^2 + [\mathcal{H}[s(t)]]^2}, \;\; \theta(t) = \arctan\left(\frac{\mathcal{H}[s(t)]}{s(t)}\right)$$

where $a(t)$ is the instantaneous amplitude and $\theta(t)$ is the phase at time t.

Then the instantaneous frequency is computed by

$$w(t) = \frac{d\theta(t)}{dt}.$$

Recently, Huang et al. [1] presented a new data analysis method, the Empirical Mode Decomposition (EMD) for analyzing nonlinear and no-stationary data. The Empirical Mode Decomposition (EMD) decomposes any signal $s(t)$ into a finite number of intrinsic mode functions(IMFs)

$$s(t) = \sum_{j=1}^{M} \varphi_j(t), \text{ where } \varphi_j(t) = a_j(t) \cos\theta_j(t).$$

Over the past decade, the EMD has gained more and more recognition. Most of the progress has been in its application. The underlying mathematical problems have been mostly left untreated. Huang listed seven outstanding mathematical problems of EMD in the book [2]. One of the outstanding problem is the data prediction problem for non-stationary processes (end effects). Since we are dealing with finite data, the algorithms must be adjusted to use some form of boundary conditions. For the EMD, the end points are problems again. And the influence of the ends will propagate into the data range during the sifting procedure. The extension of data, or data prediction, is a risky procedure even for linear and stationary processes. It's much harder for the nonlinear and nonstationary processes. Huang et al. [2] mentioned that all that is needed to be predicted for EMD are only the values and locations of the next several extrema, not all the extended data. Such a limited goal not withstanding, the task is still challenging. The traditional way to extend the data beyond the existing range for EMD is symmetric extension around the boundary.

In this work, a new method of the boundary extension is proposed. Instead of the traditional symmetric extension on the boundary, we suggest that the boundary should be extended based on the trend of the signal. Namely, we predict the signal based on the pattern of the signal. Although the signals we are dealing with are nonlinear and non-stationary, the distance between extreme points still will indicate how the frequency changes. Based on this fact, we will use the ratio of the distance between the nearest successive extreme points to predict the location of added extremal points. Also, since the signal is amplitude modulated, instead of supposing the amplitude of the following

extremal points is the same as its mirror extrema, we will use quadratic interpolation to determine the value at the added extremal points. Experimental results show that this novel idea does work better than the original EMD algorithm.

This chapter is organized as follows. In section 1.2, we describe the current EMD algorithm; in section 1.3, we define the B-spline function; in section 1.4, we present the details of the ratio boundary extension; in section 1.5, we discuss the stop criteria of the EMD algorithm; in section 1.6, we compare the numerical results of the original EMD algorithm and the new boundary extension algorithm; some conclusions are made in section 1.7.

1.2 Basic EMD

The EMD, in contrast to most of the earlier methods, works in temporal space directly rather than in the corresponding frequency space; it is intuitive, direct, and adaptive, with a posterior defined basis derived from the data. The decomposition is based on a simple assumption that any data consists of different simple intrinsic modes of oscillation. Each component is defined as an intrinsic mode function (IMF) satisfying the following conditions [1]:

(I) In the whole data set, the number of extrema and the number of zero crossings must be either equal or differ at most by one.

(II) At any data point, the mean value of the envelope defined using the local maxima and the envelope defined using the local minima is zero.

With the above definition of an IMF, one can then decompose any function through a sifting process.

In order to extract the first IMF $c_1(t)$, the component with the highest frequency embedded in the original signal $s(t)$, we using the following sifting process:

Denote by $c_1(t)$ the first IMF. Set $r_1(t) = s(t) - c_1(t)$. And repeat the sifting procedure:

$$r_2(t) = r_1(t)\text{-}c_2(t),$$
$$\ldots$$
$$r_N(t) = r_{N\text{-}1}(t)\text{-}c_N(t).$$

end when r_N has at most one extrema.

Algorithm 1.1 Sifting procedure

1. Compute a mean envelope $m_1(t)$ of the signal $s(t)$.
2. Let $h_1(t) = s(t) - m_1(t)$ be the residue.
3. If $h_1(t)$ is an IMF, STOP.

else, treat $h_1(t)$ (with its extrema) as a new signal to obtain $h_{1,1}(t)$.

4. If $h_{1,1}(t)$ is an IMF, STOP.

else, continue the same process

$$\begin{aligned} h_{1,1}(t) &= h_1(t) - m_{1,1}(t) \\ &\cdots \\ h_{1,k}(t) &= h_{1,k-1}(t) - m_{1,k}(t) \end{aligned}$$

Generally, after a finite number k_1 times, $h_{1,k_1}(t)$ will be an IMF.

Thus $s(t) = \sum_{j=1}^{N} c_j(t) + r_N(t)$ is decomposed into finitely many IMFs.

1.3 B-spline

As in the sifting procedure (algorithm 1.1), the first step is to compute the mean envelope. There are different interpolation methods to find the mean envelope. The most frequently used interpolation methods for EMD are Cubic Spline and B-spline. B-spline has some good properties which make it more suitable for EMD than other interpolation methods. In this section, we define the B-splines of order k for an arbitrary knot sequences.

For a nondecreasing sequence $\tau_j,\ j \in Z$, we define the j-th B-spline of order k as

$$B_{j,k,\tau} := (\tau_{j+k} - \tau_j)[\tau_j, ..., \tau_{j+k}](x-t)_+^{k-1},\ \ t \in \mathbb{R}$$

where $[\tau_j, ..., \tau_{j+k}]f$ represents kth divided difference of a function f at the knots $\tau_j, ..., \tau_{j+k}$.

and

$$(x-t)_+^{k-1} = \begin{cases} (x-t)^{k-1}, & \text{if } x \geq t \\ 0, & \text{if } x < 0 \end{cases}.$$

For the first order B-splines, we have

$$B_{j,1,\tau=}\begin{cases}1, & \text{if } \tau_j \le t < \tau_{j+1}\\ 0, & \text{otherwise}\end{cases}.$$

These B-splines form a basis for the space of splines of order k with knots $\tau_j,\ j \in Z$. For any function $f(t)$ in this space, $f(t)$ can be written in terms of the bases $B_{j,k,\tau}$, that is

$$f(t) = \sum_{j\in Z} \alpha_j B_{j,k,\tau}(t), \qquad t \in \mathbb{R}$$

One important property of the B-spline is its recurrence relation. For $k > 1$,

$$B_{j,k,\tau}(t) = \omega_{j,k,\tau} B_{j,k-1,\tau} + (1 - \omega_{j+1,k,\tau}) B_{j+1,k-1,\tau}$$

where

$$\omega_{j,k,\tau} = \frac{t - \tau_j}{\tau_{j+k-1} - \tau_j}$$

They are normalized so that

$$\sum_{j}^{q} B_{j,k,\tau}(t) = 1, \qquad q \ge p + k$$

When we use the B-spline to interpolate the extreme points (say $\tau = (\tau_1, \tau_2, ..., \tau_n)$) in EMD, we define the following linear functional for a function f

$$\lambda_{j,k,\tau}(f) := \frac{1}{2^{k-2}} \sum_{p=1}^{k-1} \frac{(k-1)!}{p!(k-p-1)!} f(\tau_{j+p})$$

We define

$$\widetilde{f}_{k,\tau} := \sum_{j\in Z} \lambda_{j,k,\tau}(f)\, B_{j,k,\tau}$$

as an approximation to the mean envelope in the EMD algorithm.

Particularly, when $k = 4$,

$$\widetilde{f}_{k,\tau} := \frac{1}{4} \sum_{j \in Z} \left[f(\tau_{j+1}) + 2f(\tau_{j+2}) + f(\tau_{j+3}) \right] B_{j,k,\tau}$$

which is the cubic B-spline approximation.

1.4 Boundary Extension for Nonlinear and Non-stationary Signals

The basic operation in the sifting procedure is the estimation of the mean envelope. There are two typical methods to calculate the mean envelope. One is using cubic Spline interpolation of the local maxima (respectively local minima) of $s(t)$ to get the upper envelope $U(t)$ (respectively lower envelope $L(t)$), then compute the average $m(t) = (U(t) + L(t))/2$ as the mean envelope. Another way is using the moving average of the extrema as combination of B-Spline as proposed by Chen et al. [3, 4], the mean is calculated as

$$m(t) = \sum \frac{1}{4} [s(\tau_{j+1}) + 2s(\tau_{j+2}) + s(\tau_{j+3})] B_{j,4,\tau}(t),$$

where τ_j are the local extrema point of $s(t)$.

Due to the finite observation lengths of the signal, we have to extend the extrema before we apply the Cubic Spline interpolation to find the upper/lower envelope or use B-splines to find the mean envelope. The general method is to add extrema by mirror symmetry with respect to the end points or with respect to the extrema which are closest to the end points. In this section, we will focus our attention on the following two problems:

1. How to predict the proper location of the following extrema?

2. How to predict the proper value of the extrema?

We will use the right hand end of the data to illustrate our idea. Since we are dealing with nonlinear and non-stationary signals, the frequency of the signal (i.e, the distance between two extrema) will change with time. If we use symmetric extension with respect to the last extrema, the distance between the extended two extrema is the same as the the distance between the two extrema which are closest to the end points. It works for a signal with constant frequency, but it fails when the frequency is not fixed. Let us look at the signal in

Figure 1.4.1. For this signal, we would expect that the frequency of the signal is decreasing, so the distance between two successive will become larger. But mirror extension will not predict the correct location of the added extremal points. Since the distance between the extreme reveal the frequency information, we can predict the location of the next extrem point based on the pattern of the current extremal points. Here is our strategy: we use linear approximation to estimate the change of the frequency of the signal at the end of the signal. Suppose the locations of the last three maximum points of the signal $s(t)$ are τ_{-3}, τ_{-2}, τ_{-1}. Let $r_{max} = (\tau_{-2} - \tau_{-3})/(\tau_{-1} - \tau_{-2})$ and the location of the first extended maximum be τ_1. Then the distance between any two successive maximum should keep the constant ratio r_{max}, i.e., τ_1 should satisfy $(\tau_{-1} - \tau_{-2})/(\tau_1 - \tau_{-1}) = r_{max}$. Similarly, suppose the locations of the last three minimum points of the signal $s(t)$ are η_{-3}, η_{-2}, η_{-1}. Let $r_{min} = (\eta_{-2} - \eta_{-3})/(\eta_{-1} - \eta_{-2})$ and the location of the first extended minimum be η_1. Then η_1 should satisfy $(\eta_{-1} - \eta_{-2})/(\eta_1 - \eta_{-1}) = r_{min}$. To get better result, we take the mean ratio, $r = (r_{max} + r_{min})/2$, and find τ_1, η_1 such that $(\tau_{-2} - \tau_{-3})/(\tau_{-1} - \tau_{-2}) = r$ and $(\eta_{-1} - \eta_{-2})/(\eta_1 - \eta_{-1}) = r$. Thus, we get the location of the two added maximum and two added minimum (see Figure 1.4.2(b)).

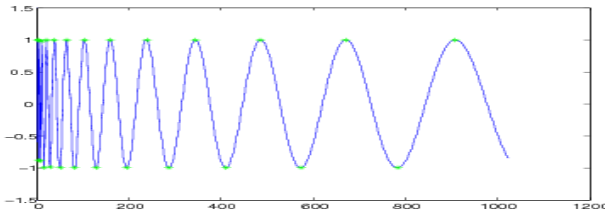
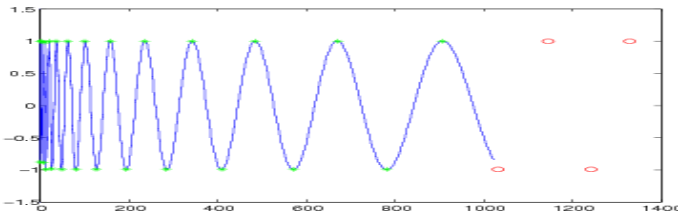
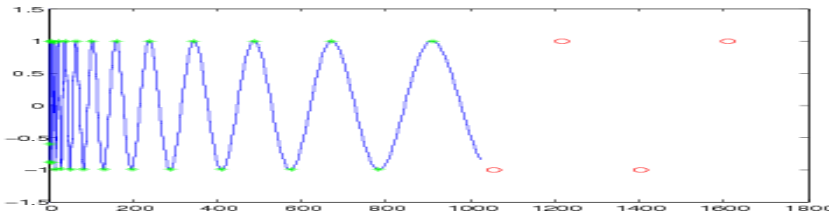


Figure 1.4.1: A frequency modulated signal

After we estimate the location of the extended extrema, we need to predict the value at the added extrema points. As mentioned in the previous section, the residual of the signal usually is not a constant. The residual shows the trend of the signal and usually it is monotonic or has one extrema. Let's look at the simplest case, one IMF + one monotonic trend as in Figure 1.4.3. If we use symmetric extension with respect to the last extrema, the extremal points will be extended as Figure 1.4.4(a). Obviously the data extension does not give the proper prediction of the original signal. It is naturally for us to think that the extrema of the signal should be extended as in Figure 1.4.4(b).



(a)



(b)

Figure 1.4.2: The circle on the graph denote the extended extrema. Fig (a) is with symmetry extension, Fig (b) is with ratio extension.

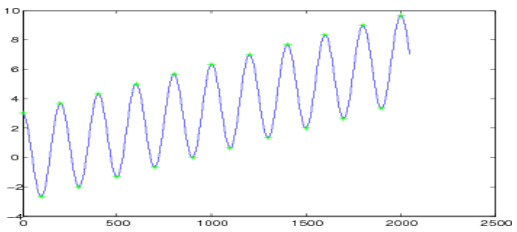
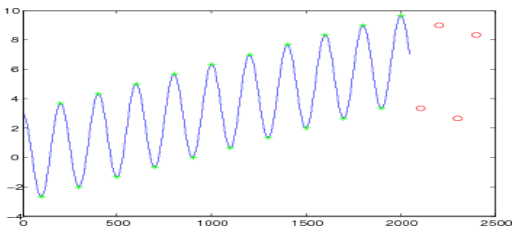


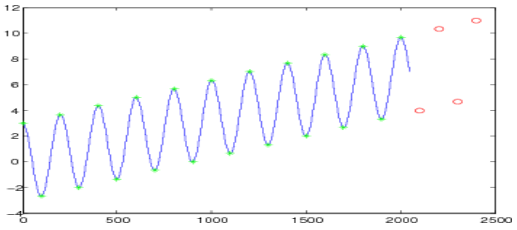
Figure 1.4.3: A signal with one sinusoid and one straight line residue

Based on this fact, we propose to predict the value of extended maximum by quadratic interpolation on the last 3 maximum of the original signal.

Combining the above mentioned ideas about predicting the location and value of the new maximum, we set up the following algorithm:



(a)



(b)

Figure 1.4.4: The circle on the graph denote the extended extrema. Fig (a) is with symmetry extension, Fig (b) is with ratio extension.

Algorithm 1.2 Boundary Extension

1. Find the locations of the last three maximum points, say τ_{-3}, τ_{-2}, τ_{-1}.
2. Calculate the ratio of the distance between the last three maximum points, $r_{max} = (\tau_{-2} - \tau_{-3})/(\tau_{-1} - \tau_{-2})$.
3. Find the locations of the last three minimum points, say η_{-3}, η_{-2}, η_{-1}.
4. Calculate the ratio of the distance between the last three minimum points, $r_{min} = (\eta_{-1} - \eta_{-2})/(\eta_1 - \eta_{-1})$.
5. Calculate the mean ratio $r = (r_{max} + r_{min})/2$.
6. Find the location (say τ_1) of the first extended maximum points , such that $(\tau_{-1} - \tau_{-2})/(\tau_1 - \tau_{-1}) = r$
7. Quadratic interpolation on the last 3 maximum of the given signal.
8. Calculate the first extended maximum using the quadratic function.

The same idea is applied to estimate the location and values of the second extended maximum points and two extended minimum points. And we extend extrema of the other side of the signal by the same method. We call this boundary extension as ***Ratio Boundary Extension*** since the distance of the extreme and the value of the extrema are proportional to the existing extrema. In most of the cases, the ratio boundary extension

works pretty well. But in some extreme situations, the ratio boundary extension may fail the maximum and minimum points interlacing condition. In that case, we will use the mirror extension about the first extreme point instead.

The results of ratio boundary extension of the signals in Figure 1.4.3, Figure 1.4.1 is shown in Figure 1.4.4(b), Figure 1.4.2(b) respectively.

1.5 Stop Criteria for Sifting

In section 2, we gave the definition of an IMF. But it's difficult to use this definition directly to evaluate whether a signal is an IMF in the numerical implementation. Thus we need to set up a stop criteria to determine whether a signal could be viewed as an IMF during the sifting procedure. The choice of stopping criteria is very important to the application of EMD, sifting too many steps may lead to loss of amplitude variation and physical meaning.

The stopping condition imposed in [1] is to limit the standard deviation computed from two consecutive results in the shifting process:

$$SD = \frac{\sum_t |h_{i,k-1}(t) - h_{i,k}(t)|^2}{\sum_t h_{i,k-1}^2(t)}$$

.

If SD is smaller than a predetermined value, the sifting process will be stopped.

Instead of using the standard deviation SD as the stop criteria, Rilling et al [5] proposed another stop criterion. For convenience, we call it **Amplitude Ratio Stop Criteria.**

Algorithm 1.3 Amplitude Ratio Stop Criteria

1. Find the upper envelope $U(t)$ and lower envelope $L(t)$
2. Introduce the mode amplitude $a(t) = [U(t) - L(t)]/2$
3. Calculate the evaluation function $\sigma(t) = |m(t)/a(t)|$
4. The sifting is stopped when $\sigma(t) < \theta_1$ for some prescribed fraction $(1 - \alpha)$ of the total duration and $\sigma(t) < \theta_2$ for the remaining fraction.

One typically set $\alpha = 0.05,\ \theta_1 = 0.05,\ \theta_2 = 10\theta_1$. This stop criteria compares the amplitude of the mean with the amplitude of the corresponding IMF. If the amplitude of

the mean envelope is relatively small compared with the amplitude of the corresponding IMF at all data points, then stop sifting. We adopted this idea to the B-spline algorithm. Since the B-spline algorithm calculate the mean envelope without calculating the upper envelope $U(t)$ and lower envelope $L(t)$, instead of calculating the mode amplitude $a(t) = [U(t) - L(t)]/2$, we use B-Spline again and let

$$\begin{aligned} a(t) &= \sum \tfrac{1}{4}[|\, s(\tau_{j+1}) - s(\tau_{j+2})| + |s(\tau_{j+2}) - s(\tau_{j+3})\,|]B_{j,4,\tau}(t) \\ &= \sum \tfrac{1}{4}|\, s(\tau_{j+1}) - 2s(\tau_{j+2}) + s(\tau_{j+3})\,|B_{j,4,\tau}(t) \end{aligned}$$

where τ_j are the local extrema point of $s(t)$.

The experiments show that the mode amplitude function $a(t)$ defined by the B-Spline function works similarly to the difference between the upper envelope and lower envelope. The reason that we want to adopt the Amplitude Ratio Stop Criteria is that it is aimed at guaranteeing globally small fluctuations in the mean while taking into account locally large excursions.

1.6 Numerical Results

In this section, we will compare the decomposition results of some examples by the modified B-spline method with ratio boundary extension proposed in section 3 and Amplitude Ratio Stop Criteria as in section 4. We will compare the results with the original B-spline method by Chen et al. [3], the original Cubic Spline method by Flandrin et al. [6, 5] and the modified Cubic Spline with ratio boundary extension.

Example 1. $s(t) = 10 * \cos((\frac{t}{80})^{1.5} * \pi) + 2 * \cos((\frac{t}{100})^{0.8} * \pi) + (\frac{t}{500} + 2), \;\; t = 20 : 2^{11}.$

Figure 1.6.1 is the graph of the original signal and its components. Ideally, this signal should be decomposed as 2 IMFs, $(10 * \cos((\frac{t}{80})^{1.5} * \pi),\; 2 * \cos((\frac{t}{100})^{0.8} * \pi))$ and a residue $(\frac{t}{500} + 2)$. The two IMFs are Frequency Modulated signals. Figure 1.6.2 shows the decomposition results of the four different methods. From the graph, we find that due to the boundary effect, the original B-spline method and the original Cubic Spline method can not decompose the signal correctly from the second IMF, and we can not get the correct residue. The modified Cubic Spline with the ratio boundary extension and the modified B-spline method with ratio boundary extension and amplitude ratio stop criteria decompose the signal almost perfectly.

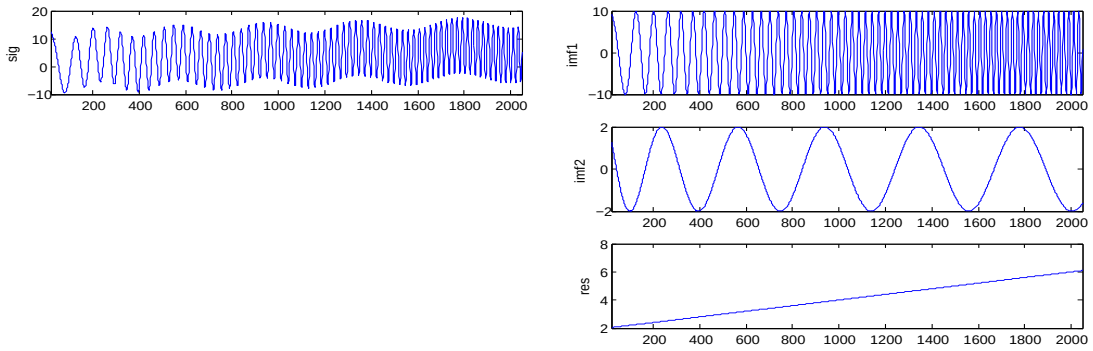


Figure 1.6.1: Graph for Example 1: a signal with two FM components and a linear residue

Example 2. : $s(t) = 5 * \cos((\frac{t}{120})^{1.8} * \pi) + (\frac{t}{200})^{1.5} * \cos(\frac{t}{180} * \pi) + (\frac{t}{200} - 5)^2, \quad t = 20 : 2^{11}.$

Figure 1.6.3 is the graph of the original signal and its components. This signal has a frequency modulated component $5 * \cos((\frac{t}{120})^{1.8} * \pi)$, an amplitude modulated component $(\frac{t}{200})^{1.5} * \cos(\frac{t}{180} * \pi)$ and a quadratic residue $(\frac{t}{200} - 45)$. Again, if we compare the result of the four different algorithm as in Figure 1.6.4, we find that the original B-spline method and the original Cubic Spline method over-decompose the signal, the right hand side of the second IMF is bad due to the improper boundary extension and the residue do not show any trend of the original signal. The modified B-spline method with ratio boundary extension and amplitude ratio stop criteria, and the modified Cubic Spline method with amplitude ratio stop criteria decompose the signal as two IMFs and one residue as expected.

Based on the numerical results in the above examples, we conclude that the ratio boundary extension and Amplitude Ratio Stop Criteria indeed give us an improved implementation of the Empirical Mode Decomposition.

1.7 Discussions

The Empirical Mode Decomposition (EMD) is a promising tool for the analysis of non-stationary and nonlinear signal processing. It has been applied with great success for nonlinear and nonstationary signal analysis in various areas.

This chapter propose some new ideas on the EMD algorithm. We propose the ratio boundary extension which is more adaptive to the signal compared with the symmetric extension. We would like to emphasize the importance of the boundary extension. As

mentioned in other papers, the influence of the ends will still propagate into the data range in the low frequency components. And from the numerical experiments, we find that the proper data prediction is important for us to get the correct IMFs. If we make a wrong prediction at the first step, the whole signal will be decomposed incorrectly from the first step and it will ruin all the following EMD decomposition steps. So it is important to choose a proper boundary extension at every step. We also investigate the stop criteria and applied the Amplitude Ratio Stop Criteria to our B-Spline algorithm.

Up to now, most of the progress with EMD are in its application. The decomposition is only defined as the output of an algorithm. The mathematical ground for EMD has not been set up yet. In the future, we will devote to the theoretical research on EMD.

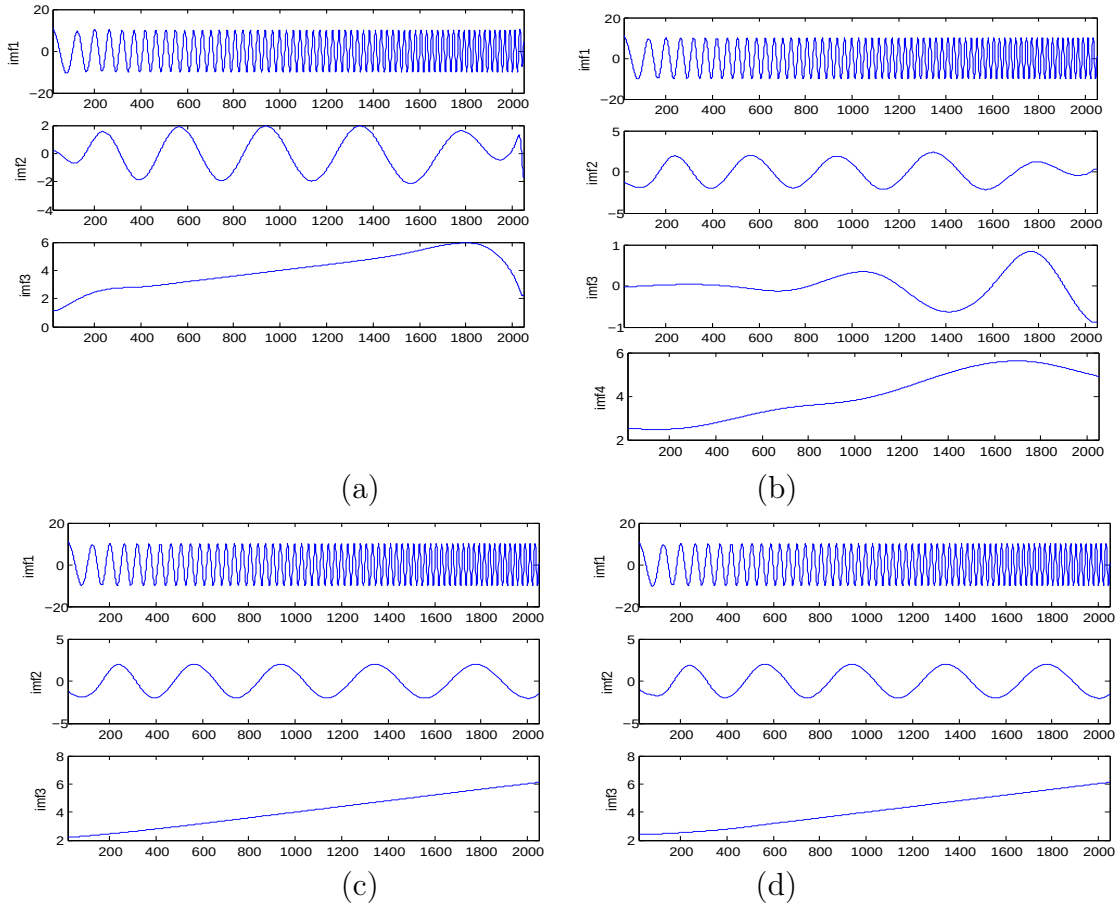


Figure 1.6.2: Fig(a) is the IMFs of Example 1 by the original B-Spline method with symmetric boundary extension and original stop criteria; Fig(b) is the IMFs by Cubic Spline method with symmetric boundary extension and amplitude ratio stop criteria; Fig(c) is the IMFs by B-Spline method with ratio boundary extension and amplitude ratio stop criteria; Fig(d) is the IMFs by Cubic Spline with ratio boundary extension and amplitude ratio stop criteria.

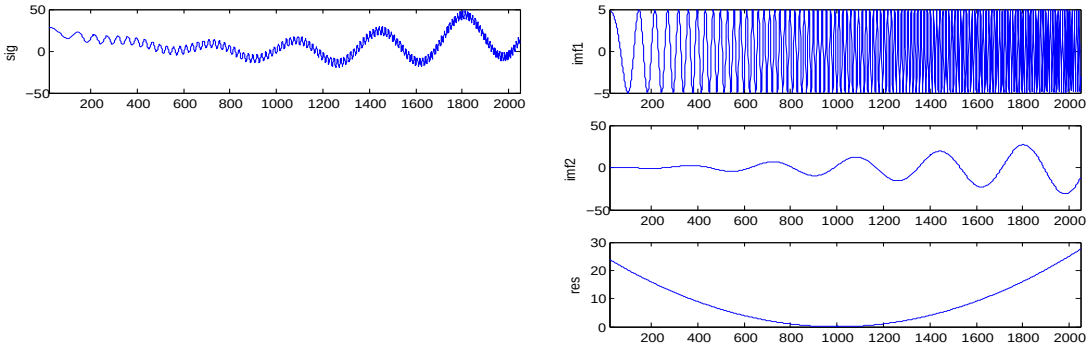


Figure 1.6.3: Graph for Example 2: A signal with one FM component, one AM component and a linear residue

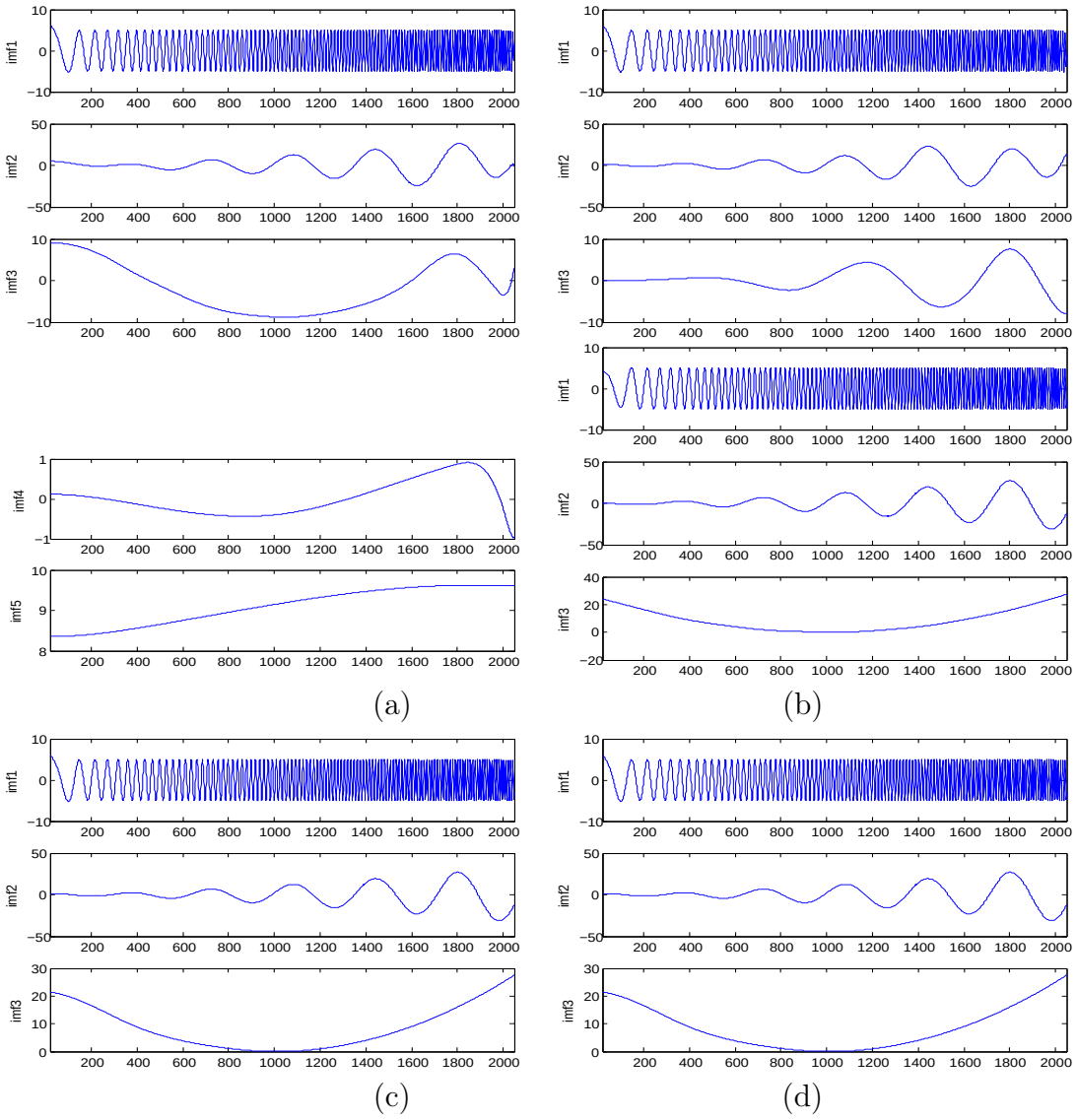


Figure 1.6.4: Fig(a) is the IMFs of Example 2 by the original B-Spline method with symmetric boundary extension and original stop criteria; Fig(b) is the IMFs by Cubic Spline method with symmetric boundary extension and amplitude ratio stop criteria; Fig(c) is the IMFs by B-Spline method with ratio boundary extension and amplitude ratio stop criteria; Fig(d) is the IMFs by Cubic Spline with ratio boundary extension and amplitude ratio stop criteria.

Chapter 2

Graph Model for Pattern Recognition in Text

2.1 Introduction

For text archives containing a large number of documents, determining the similarity of documents is an area of research that has seen a great deal of activity in recent years. With the advent and ubiquity of internet communication the search for related documents plays an important role in such applications as search, detection of fraud and the detection of conspiring groups. Term frequency has long been used as a tool for estimating the probabilistic distribution of features in a document. A number of applications have been developed including language modeling [7], feature selection [10, 11] and term weighting [12, 13]. Based on the term frequency information, documents can be classified by several clustering methods such as decision trees [15], neural networks [8, 16], Bayesian methods [17, 18] or support vector machines [9, 19, 14].

The term frequency method is an effective approach if a rough classification of documents based on their subjects or themes. However, if one would like to further determine the similarity of writing patterns or determine the authorship of documents, the traditional term frequency method will provide only very rough estimates with little accuracy or reliability. The main drawback of the term frequency method is the fact that it relies on a bag-of-words [20, 21, 22] approach. It implies feature independence, and disregards any dependencies that may exist between words in the text. The bag-of-words model may not

be the best technique to capture keyword importance. If the text structure information could be preserved properly at the same time, it would lead to a better keyword weighting scheme [23].

In this chapter, we introduce a new approach that exploits not only the keyword frequency but also their location and ordering. We represent a document as a weighted directed multigraph by taking keywords as the vertices and constructing arcs whose weighting contains the relation information of a keyword to other keywords. The adjacency matrix of the graph induces a signature vector for the document. A clustering method is then applied to the set of signature vectors for grouping similar documents into clusters. With this new approach, we are able to evaluate the similarity between any two documents from a set of text documents within the SAME category.

A set of detailed algorithms for the estimation of signature vectors and clustering are presented in this work. This algorithm has been applied to two sets of sample documents.

1. Nigerian Fraud Emails, each of which has the same topic: to transfer money into some bank accounts in order to receive lager sum of payback.

2. Papers in academic journals in graph theory, some of which are known to be plagiarized.

Each group is in one category, and therefore, keywords may appear with similar frequencies. The traditional method of sorting documents by keyword-frequency is able to filter this group out off a lager subset of documents with many different subjects. However, by considering the ordering and location of keywords, we are able to further evaluate their similarity within their own group, i.e. to classify fraudulent emails authored by the same person or copy-pasted types with slight modification, or to identify the plagiarized papers.

In next section, we describe the schema for representing a document as a weighted directed multigraph. Section 2.3 discusses the computation complexity. In section 2.4, we present some application examples of our algorithm. Finally, in section 2.5, some discussions are presented and future research problems are outlined.

2.2 Graph Model for Pattern Recognition

The overall approach of this algorithm begins with the identification of a set of relevant keywords. Once these are selected, we then aggregate the relative distances of the key-

words with a document. This in turn is used to construct a weighted directed multigraph that generates representing vectors for each document in a high dimensional feature space. These vectors can then be used to determine similarity values for any pair of documents.

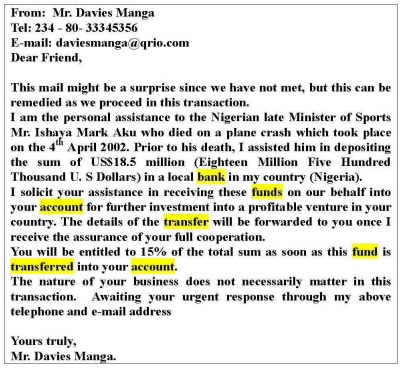
2.2.1 Summary of our Method

Step 1: Using a weighted directed multigraph to find a signature vector for each document.
Step 2: Calculate the similarities between any two documents via their signature vectors.
Step 3: Using Quasi-Clique Merge clustering method to classify all documents.

We will explain the details of each step by a simple example.

2.2.2 Details of the Step 1

To have a clear view of the algorithm, we will use the example illustrated in Figure 2.2.1 to explain the procedure [24].



From: Mr. Davies Manga
Tel: 234 - 80- 33345356
E-mail: daviesmanga@qrio.com
Dear Friend,

This mail might be a surprise since we have not met, but this can be remedied as we proceed in this transaction.
I am the personal assistance to the Nigerian late Minister of Sports Mr. Ishaya Mark Aku who died on a plane crash which took place on the 4th April 2002. Prior to his death, I assisted him in depositing the sum of US$18.5 million (Eighteen Million Five Hundred Thousand U. S Dollars) in a local bank in my country (Nigeria).
I solicit your assistance in receiving these funds on our behalf into your account for further investment into a profitable venture in your country. The details of the transfer will be forwarded to you once I receive the assurance of your full cooperation.
You will be entitled to 15% of the total sum as soon as this fund is transferred into your account.
The nature of your business does not necessarily matter in this transaction. Awaiting your urgent response through my above telephone and e-mail address

Yours truly,
Mr. Davies Manga.

Figure 2.2.1: A fraudulent email

2.2.2.1 Record the keyword information appeared in the document.

For a given document, the following steps are applied to it. Suppose we have already chosen a set of words as keywords, say $K = \{\overline{K}_1, \overline{K}_2, ..., \overline{K}_m\}$. Record every keyword and its position in the document. We will use the following notation:

- k_i represents one of the keyword in the keyword set K .

- i represents the order of the keyword appearing in the document. (It is possible that k_i, k_j are the same element in K.)
- $\widetilde{m}$ represents the total number of keywords appearing in the document .
- p_i is a integer, which represents the total number of the words from the beginning of the document to the word k_i.

In addition we record the frequency of each keyword at the same time. Thus we have the Keyword-Position information table(Table 2.1).

Keyword appears in the document	Position in the document
k_1	p_1
k_2	p_2
$\vdots$	$\vdots$
$k_{\widetilde{m}}$	$p_{\widetilde{m}}$

Table 2.1: Keyword-Position table

The details of this process are illustrated as follows (with Figure 2.2.1 as an example). For this example, we use the keyword set: $\{bank,\ fund,\ account,\ transfer\}$. Its Keyword-Position information is listed in Table 2.2. Frequency information of each keyword for the given example (Figure 2.2.1) is listed in Table 2.3

Keyword appears in the document	Position in the document
bank	91
fund	103
account	109
transfer	124
fund	153
transfer	155
account	158

Table 2.2: Keyword-Position information of the email in Figure 2.2.1

keyword	frequency
bank	1
fund	2
account	2
transfer	2

Table 2.3: Keyword-Frequency information of the email in Figure 2.2.1

2.2.2.2 Construct a weighted directed multigraph

For a given document D and a set of keywords K, let G_m be a weighted directed multigraph G_m with the vertex set $K = \{\overline{K}_1, \overline{K}_2, ..., \overline{K}_m\}$. constructed as follows.

Suppose that $k_1, \cdots, k_s$ is the sequence of words such that
(1) each k_μ is a keyword of the given set K,
(2) $k_1, \cdots, k_s$ appear in the document D in this order,
(3) the position of the word k_μ in the document D is p_μ (the p_μ-th word in the document D, $(1 \leq p_1 < \cdots < p_s)$.

Add an arc from the vertex k_i to the vertex k_j with the weight $w\_m_{ij} = p_j - p_i + 1$, which is the distance from the word k_i to the word k_j in the document D.

Note that if k_i and k_j are the same element of the set K, they are the same vertex in the graph.

A large weight for a given arc indicates that the corresponding pair of keywords are relatively far away from each other and, therefore, their logical connection are relatively "weak" in the document. Thus, we may ignore those arcs with large weights. (We choose a $threshold = 200$ in our example in Figure 2.2.1 and delete any arc with weight greater than 200.)

Note that the resulted weighted directed multigraph may contain not only parallel arcs but also loops.

For the given example (Figure 2.2.1), its corresponding weighted directed multigraph is Figure 2.2.2.

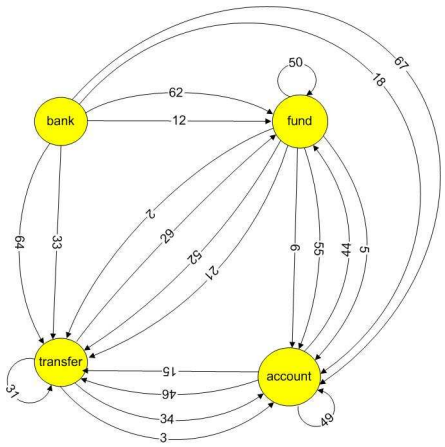


Figure 2.2.2: The weighted, directed multigraph of the email in Figure 2.2.1

2.2.2.3 Simplification of representing graphs

The weighted directed multigraph G_m constructed in the previous step is further simplified as follows (a directed graph G_s is constructed from G_m, in which, parallel arcs are combined).

Let $E_{ij} = \{k_\mu k_\nu |\ \ k_\mu = \overline{K_i}\ \&\ k_\nu = \overline{K_j}\}$, which is the set of all arcs from the vertex $\overline{K_i}$ to the vertex $\overline{K_j}$ of the weighted directed multigraph G_m.

Let $K = \{\overline{K}_1, \overline{K}_2, ..., \overline{K}_m\}$. be the vertex set of the new directed graph G_s. For each pair of vertices $\overline{K_i}$ and $\overline{K_j}$ ($i, j = 1, 2, ..., m$), if $E_{ij} \neq \emptyset$, put an arc e_{ij} from $\overline{K_i}$ to $\overline{K_j}$. The weight of the arc $e_{ij} = \overline{K_i K_j}$ is calculated as follows,

$$w\_s_{ij} = \sum_{k_\mu k_\nu \in E_{ij}} \frac{1}{w\_m_{\mu\nu}}, \qquad \text{if } E_{ij} \neq \emptyset.$$

The terms $\frac{1}{w\_m_{\mu\nu}}$ are constructed so that when two terms are closer to each other the reciprocal of their small relative distance will contribute more strongly to the summation. When terms are farther apart, the reciprocal will be small and so these terms will contribute less.

The simplified directed graph G_s of the given example is illustrated in Figure 2.2.3.

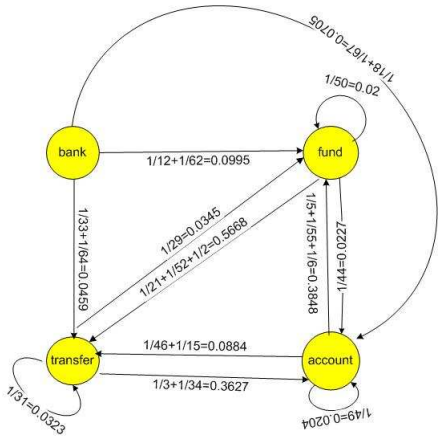


Figure 2.2.3: the simplified directed graph of the email in Figure 2.2.1

2.2.2.4 Create a Signature Vector to Represent the Input Email.

Now we create a signature vector to represent an input email by the frequency information of the keywords and the simplified weighted graph information.

1. We use f_i to denote the frequency of the keyword $\overline{K_i}$ in the document. Use $F(D) = [f_1, f_2, ..., f_m]$ denote the frequency vector of the document D.

2. We use the adjacency matrix to represent the simplified weighted directed graph G_s.

Let $w\_s_{ij} = 0$ if there is no arc from the vertices $\overline{K_i}$ to $\overline{K_j}$.

$$W(D) = \begin{bmatrix} w\_s_{11} & w\_s_{12} & \cdots & w\_s_{1m} \\ w\_s_{21} & w\_s_{22} & \cdots & w\_s_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ w\_s_{m1} & w\_s_{m2} & ... & w\_s_{mm} \end{bmatrix}.$$

Then we rewrite it as an $(m \times m)$ vector.

$$\widetilde{W}(D) = [w\_s_{11}, w\_s_{12}, ..., w\_s_{1m}, w\_s_{21}, w\_s_{22}, ...w\_s_{2m}, ..., w\_s_{mm}].$$

Let $R(D) = [F(D), \widetilde{W}(D)]$. The vector $R(D)$ contains not only the frequency information of the keywords, but also the structure information of the document. It is used as the signature vector of the document.

Again, corresponding to the given example (Figure 2.2.1), we have

$$F(D) = [1, 2, 2, 2]$$

$$W(D) = \begin{bmatrix} 0 & 0.0995 & 0.0705 & 0.0459 \\ 0 & 0.0200 & 0.3848 & 0.5668 \\ 0 & 0.0227 & 0.0204 & 0.0884 \\ 0 & 0.0345 & 0.3627 & 0.0323 \end{bmatrix}$$

$$R(D) = \begin{array}{l} [1,\ 2,\ 2,\ 2,\ 0,\ 0.0995,\ 0.0705,\ 0.0459,\ 0,\ 0.0200,\ 0.3848,\ 0.5668, \\ 0,\ 0.0227,\ 0.0204,\ 0.0884,\ 0,\ 0.0345,\ 0.3627,\ 0.0323] \end{array}.$$

2.2.3 Details of the Step 2

2.2.3.1 Find Signature Vectors for all Documents

Repeat the process of the Step 1, we create signature vectors for all documents.

Let $R(D_i)$ be the signature vector of the i-th document.

Let $M = [R(D_1),\ R(D_2),\ ...,\ R(D_{n-1}),\ R(D_n)]^T$, then M is an $n \times (m + m^2)$ matrix (n is the total number of the documents, m is the cardinality of the keywords set K. Each row of the matrix represents a document.

2.2.3.2 Normalization of the Matrix

We normalize the matrix M with respect to the columns for the purpose of the compatibility in every dimension. We denote the normalized matrix as

$$\widetilde{M} = [\widetilde{R}(D_1),\ \widetilde{R}(D_2),\ ...,\ \widetilde{R}(D_{n-1}),\ \widetilde{R}(D_n)]^T.$$

And the details of the normalization is presented in next section.

2.2.3.3 Similarity

The similarity S_{ab} between any two documents $\overline{D}_a, \overline{D}_b$ is determined by the cosine similarity as follows

$$S_{ab} = \frac{|\widetilde{R}(D_a) \cdot \widetilde{R}(D_b)|}{|\widetilde{R}(D_a)| \cdot |\widetilde{R}(D_b)|}$$

where $\widetilde{R}(D_a),\ \widetilde{R}(D_b)$ are the normalized signature vectors of the documents $\overline{D}_a, \overline{D}_b$.

2.2.4 Details of the Step 3

A variety of different clustering algorithms have been developed and implemented in popular statistical software packages. A general review of cluster analysis can be found in many references, for instance, [27, 28, 29], etc. None of these algorithms can, in general, rigorously guarantee to produce a globally optimal clustering for non-trivial objective functions [30].

After calculating the pairwise similarities of all documents, we then classify these documents into different groups by applying the Quasi-Clique Merge(QCM) method to cluster the documents. It is observed that one of the most significant differences between the QCM method and other clustering algorithms is that the QCM method constructs a much smaller hierarchical tree. This tree structure leads to better identification of meaningful clusters since there are fewer subdivisions of the data set due to the impact of irrelevant or improperly interpreted information. Additionally, the QCM method results in multi-membership clustering [31], which preserves some amount of the ambiguity inherent in the data set rather than errantly suppressing it as many other clustering algorithms do.

2.3 The Algorithm and Complexity Analysis

2.3.1 Graph Theory Notation and Terminology

Let Σ be the set of the alphabets appearing in the key words which includes the special symbol "$\square$" as the space character.

Let $\mathcal{D} = \{\overline{D_1}, \overline{D_2}, \cdots, \overline{D_n}\}$ be a set of text documents for pattern detection. Each document $\overline{D_i}$ is a sequence $d_{i,0} \cdots d_{i,t_i}$ consisting of alphabets from the set Σ, where the first and the last character $d_{i,0} = d_{i,t_i} = \square$, $t_i + 1$ is the length of the document $\overline{D_i}$.

Let $K = \{\overline{K}_1, \overline{K}_2, ..., \overline{K}_m\}$. be the set of selected keywords. For each keyword $\overline{K_i} = k_{i,0} \cdots k_{i,s_i}$, the first and the last character $k_{i,0} = k_{i,s_i} = \square$, $s_i + 1$ is the length of

the keyword $\overline{K_i}$.

Let $G = (V, A)$ be a directed graph with vertex set V and arc set A.

$N^+(v)$ is the set of all out-neighbors of the vertex v. That is, $N^+(v) = \{u \in V(G) : vu \in A(G)\}$.

$N^-(v)$ is the set of all in-neighbors of the vertex v. That is, $N^-(v) = \{u \in V(G) : uv \in A(G)\}$.

Let $L : A(G) \mapsto \Sigma$ be a labeling of $A(G)$. $L^+(v) = \{l(vu) : u \in N^+(v)\}$.

2.3.2 Construction of Searching tree

For the purpose of finding keywords efficiently, we use the following algorithm to set up a searching tree for keyword searching.

2.3.2.1 Algorithm

Algorithm 2.1 Construction of Searching tree

Input. $K = \{\overline{K}_1, \overline{K}_2, ..., \overline{K}_m\}$.: a set of keywords.

Output. A rooted tree (called "searching tree") T: T has a root v_0 and m leaves. Each of the leaf represents a keyword; each arc of T is labeled with a character $\in \Sigma$; for each leaf v_ℓ, let P be the unique directed path from the root v_0 to v_ℓ, the sequence of labels along the path P coincides with characters of the keywords K_ℓ. (Figure 2.3.1 gives a simple example of a searching tree.)

Initial step. T has a root v_0 and a vertex v_1, and an arc v_0v_1 with the label $L(v_0v_1) = \square$.
$i \leftarrow 1$: i is the keyword index (current keyword $\overline{K_i} = k_{i,0} \cdots k_{i,s_i}$ that is under processing, $k_{i,0} = k_{i,s_i} = \square$, $s_i + 1$ is the length of the keyword $\overline{K_i}$.)
$\lambda \leftarrow 1$: λ is the level index (the character $k_{i,\lambda}$ is currently under processing, and λ is also current level of the tree that is under construction).
$v \leftarrow v_1$: (the current vertex whose out-neighborhood is under construction.)

Step 1.
Case 1. If $\lambda < s_i$. Consider $N^+(v)$.
Subcase 2-a. If $N^+(v) = \emptyset$, or if $k_{i,\lambda} \notin L^+(v)$,
then go to Step 2.
Subcase 2-b. If $k_{i,\lambda} \in L^+(v)$,
say, $k_{i,\lambda} = L(vu)$ for some $u \in N^+(v)$,
then go to Step 3.
Case 2. If $\lambda = s_i$. (Reach the end of the keyword $\overline{K_i}$.)
If $i < m$ then
$i \leftarrow i + 1$
$\lambda \leftarrow 1$
$v \leftarrow v_1$
and go back to Step 1;
If $i = m$ (reach the last keyword) then go to Step F.

Step 2. (This is the step that adds a directed path with tail at the vertex v).
Add a directed path $u_0 \cdots u_z$ with $\{u_1, \cdots, u_z\}$ as new vertices and $u_0 = v$ and

$$l(u_0u_1) = k_{i,\lambda},\ l(u_1u_2) = k_{i,\lambda+1},\ \cdots,\ l(u_{z-1}u_z) = k_{i,s_i}.$$

Then
$\lambda \leftarrow s_i$
and go to Step 1.

Step 3. (In this step, an existing arc vu will be used since $l(vu) = k_{i,\lambda}$).
$\lambda \leftarrow \lambda + 1$,
$v \leftarrow u$,
go to Step 1.

Step F. Final step: Output.

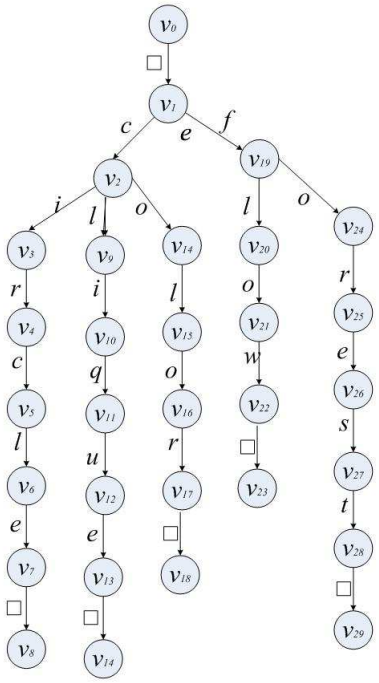


Figure 2.3.1: A searching tree with keyword {circle, clique, color, flow, forest}

2.3.2.2 Complexity

Let $len_K = \sum_{i=1}^{m} |\overline{K_i}|$ denote the total length of all keywords in K. Steps 1 - 3 form a loop that repeats len_K times. For Case 1, and Subcase 2-a, each costs 1 unit for each character of $\overline{K}_i$; for Subcases 2-b, it costs at most $|\Sigma|$ units (for comparisons). For each subcase, an iteration of Step 2 or 3 is followed and afterward, return back to Step 1 for another loop.

Hence, the complexity of constructing the searching tree is $O(len_K)$.

Remark: since we only build up this tree once in the whole procedure, the complexity of constructing the searching tree will not be counted into the total complexity.

2.3.3 Keyword Searching and Location in Documents

2.3.3.1 Algorithm

Let $K = \{\overline{K}_1, \overline{K}_2, ..., \overline{K}_m\}$. be the set of selected keywords. A keyword searching tree T was constructed in Algorithm 2.1 ready for use. Let $\Theta(T)$ be the set of leaves of the rooted tree T. For the sake of convenience, each leaf of T is denoted by its corresponding keyword. That is, $\Theta(T) = K = \{\overline{K}_1, \overline{K}_2, ..., \overline{K}_m\}$.

2.3.3.2 Complexity

Each character $d_{i,j}$ of the document $\overline{D_i}$ is compared with $N^+(v)$ or $N^+(v) \cup N^+(v_0)$ for some vertex $v \in V(T)$. That is, it costs at most $(|\Sigma| + 1)$ units for comparisons. So, the total cost is $(|\Sigma| + c) \times t_i$ where c is a small constant cost for re-indexing of j, v and updating the records $P(\overline{K}_\mu)$.

Thus, the complexity of keyword searching is $O(t_i)$, where t_i is the length of the input document D_i.

2.3.4 Signature Vector of a Document

The *signature vector* $R(\overline{D_i})$ for a given document $\overline{D_i}$ is to be calculated in this section.

Input: The collection of sets $P(\overline{K}_\mu)$ for all keyword $\overline{K}_\mu$ (provided in Algorithm 2.2).
Output: An $1 \times (m + m^2)$-vector $R(\overline{D_i})$.

Calculation:

Let $F(\overline{D_i}) = [f_\mu] = [f_1, \cdots, f_m]$ be a $(1 \times m)$-matrix where $f_\mu = |P(\overline{K}_\mu)|$.
Let $W(\overline{D_i}) = [\alpha_{\mu,\nu}]$ be an $(m \times m)$-matrix with

$$\alpha_{\mu,\nu} = \sum \frac{1}{p_\mu - p_\nu + 1}$$

where the summation is over all pairs $p_\mu \in P(\overline{K}_\mu)$ and $p_\nu \in P(\overline{K}_\mu)$ with $p_\mu > p_\nu$.

Note: this is not a symmetric matrix, parallel arcs in opposite directions in the graph are considered differently.

Algorithm 2.2 Keyword Searching and Location in Documents

Input. A text document $\overline{D_i} = d_{i,0} \cdots d_{i,t_i}$ where the first and the last character $d_{i,0} = d_{i,t_i} = \square$.

Output. The position sets of each keyword in the document. Each keyword $\overline{K}_\mu$ is associated with a set $P(\overline{K}_\mu)$ of integers, where: $p \in P(\overline{K}_\mu)$ if and only if the keyword $\overline{K}_\mu$ appears in the document $\overline{D_i}$ at position p.

Initial Step. $j \leftarrow 0$ (the character $d_{i,j}$ of the document $\overline{D_i}$ is currently in iteration).
$v \leftarrow v_0$
$P(\overline{K}_\mu) \leftarrow \emptyset$, for each $\overline{K}_\mu$.
$w \leftarrow 1$: w is the position of current word in the document.

Step 1. If $j < t_i$, go to Step 2.
If $j = t_i$, go to Step F.

Step 2.
If $d_{i,j} \in L^+(v)$, say $l(vu) = d_{i,j}$ where $u \in N^+(v)$, then go to Step 3.
If $d_{i,j} \notin L^+(v)$, then go to Step 4.

Step 3.
If $N^+(u) \neq \emptyset$ (u is not a leave of the tree T), then
$v \leftarrow u$,
$j \leftarrow j + 1$,
go to Step 1.

If $N^+(u) = \emptyset$ (u is a leave of the tree T), then
$P(u) \leftarrow P(u) \cup \{w\}$,
$v \leftarrow v_0$,
$j \leftarrow j + 1$,
$w \leftarrow w + 1$,
go to Step 1.

Step 4.
Case 1. If $d_{ij} \neq \square$,
$j \leftarrow j + 1$
and go back to Step 4;

Case 2. If $d_{ij} = \square$,
$v \leftarrow v_0$,
$w \leftarrow w + 1$,
Go to Step 1.

Step F. Output: $P(\overline{K}_\mu)$, for each $\overline{K}_\mu \in V_L$.

Let $R(\overline{D_i}) = [F(\overline{D_i}), \widetilde{W}(\overline{D_i})]$. Here we rewrite the matrix $W(\overline{D_i})$ as a $1 \times m^2$ row vector $\widetilde{W}(\overline{D_i})$.

Complexity: It costs $|P(\overline{K}_\mu)|$ for the calculation of f_μ, and it costs $|P(\overline{K}_\mu)||P(\overline{K}_\nu)|$ for the calculation of $\alpha_{\mu,\nu}$ and $\alpha_{\nu,\mu}$ for every $\mu, \nu \in \{1, \cdots, m\}$. So, the complexity is $O(m^2\phi^2)$, where ϕ = average of $|P(\overline{K}_\mu)||$ (the average appearance of a keyword in the document $\overline{D_i}$).

2.3.5 Similarity Calculation

Let $D = \{\overline{D}_1, \cdots, \overline{D}_n\}$ be a set of documents.

2.3.5.1 Date Normalization

Input: Let $R(\overline{D_i})$ be the $(1 \times (m + m^2))$ vector calculated in Section 2.3.4. Additionally, let $M(D) = [\beta_{i,j}]$ be the $(n \times (m + m^2))$-matrix, with the i-th row of the signature vector $R(\overline{D_i})$, so that $\beta_{i,j}$ is the j-th component of the signature vector $R(\overline{D_i})$.

Calculation and output: For each $j \in \{1, \cdots, m + m^2\}$, let A_j be the average of all cells in the j-th column of the matrix $M(D)$.

$$\widetilde{M}(D) := [\widetilde{\beta}_{i,j}] = [\frac{\beta_{i,j}}{A_j}].$$

Complexity: $O(nm^2)$

2.3.5.2 Similarity

The similarity between two documents $\overline{D}_a$ and $\overline{D}_b$ is then calculated as

$$s_{ab} = \frac{\widetilde{R}(\overline{D}_a) \cdot \widetilde{R}(\overline{D}_b)}{|\widetilde{R}(\overline{D}_a)| \cdot |\widetilde{R}(\overline{D}_b)|}$$

where $\widetilde{R}(\overline{D_i})$ is the i-th row of the matrix $\widetilde{M}(D)$, namely, the normalized signature vector of the document $\overline{D}_i$.

Complexity: $O(n^2m^2)$.

2.3.6 Clustering

The final stage is to use Quasi-Clique Merge algorithm (QCM [31]) to cluster all documents. We suppose h is the level number of the hierarchical system. Then, by the estimation in [31], the number of iterations is bounded by $O(hn^2log(n))$. Note that, for an input set of n documents, the number of hierarchical levels is $\log(n)$ in average. Thus, the complexity of QCM is $O([nlog(n)]^2)$.

2.3.7 Total Complexity

By summing up all steps, the total complexity is

$O(t + m^2\phi^2 + nm^2 + n^2m^2 + [nlog(n)]^2)$,

where $|\Sigma|$ is the number of the distinct alphabets appearing in the key words, t is the average length of the documents, ϕ is the average appearance of a keyword in a document, m is the total number of keywords, n is the total number of documents.

Since we compare lots of documents, ϕ (the average appearance of a keyword in a document) is much smaller than n (the total number of documents), and t is usually less than n^2m^2. Thus, the complexity is further simplified as $O(n^2m^2 + [nlog(n)]^2)$.

2.4 Experimental Results

In order to evaluate the effectiveness of our algorithm, we will compare the results of our method with the usual keyword frequency method. We calculate the similarity between every pair of documents the following two different ways:
KF method: *only use keyword frequency information.*
KFP method: *use keyword frequency and structure information, which is based on the weighted directed multigraph model described in this work.*

In the following analysis we will show that the KFP method is superior to the KF method.

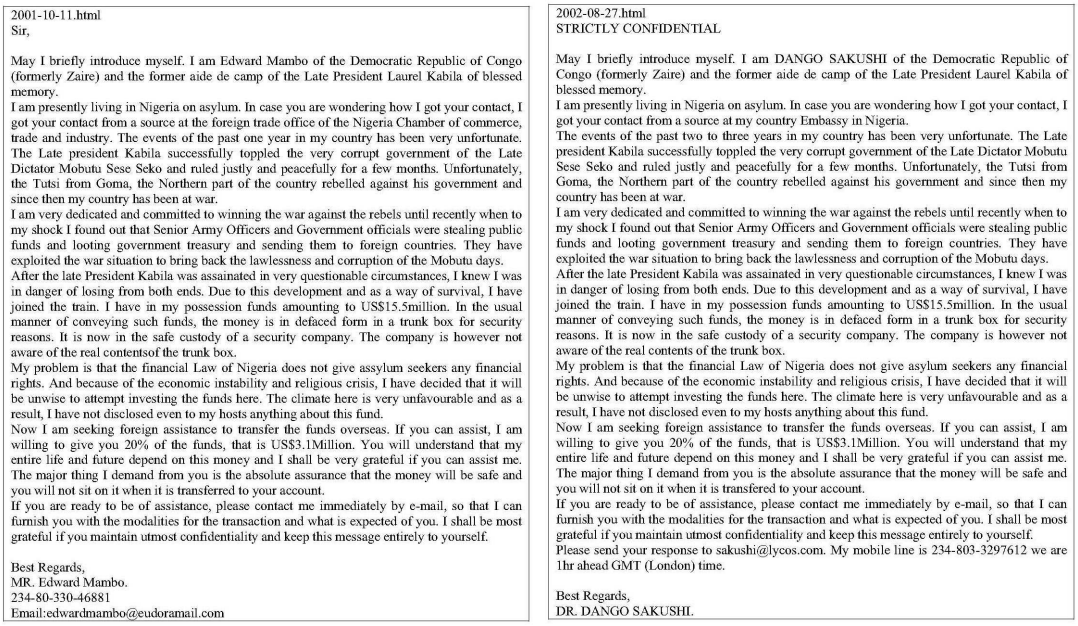
2.4.1 Nigerian Fraud Emails

We acquired 542 different Nigerian Fraud Emails from an internet archive [24]. We wish to cluster these emails in order to determine any commonality in the authorship of the

texts.

In the following experiment, we choose $\{bank, account, money, fund, business, transaction\}$ as the keyword set. Consider two emails: *2001-10-11.html*, *2002-08-27.html* (Figure 2.4.1)

.



2001-10-11.html
Sir,

May I briefly introduce myself. I am Edward Mambo of the Democratic Republic of Congo (formerly Zaire) and the former aide de camp of the Late President Laurel Kabila of blessed memory.
I am presently living in Nigeria on asylum. In case you are wondering how I got your contact, I got your contact from a source at the foreign trade office of the Nigeria Chamber of commerce, trade and industry. The events of the past one year in my country has been very unfortunate. The Late president Kabila successfully toppled the very corrupt government of the Late Dictator Mobutu Sese Seko and ruled justly and peacefully for a few months. Unfortunately, the Tutsi from Goma, the Northern part of the country rebelled against his government and since then my country has been at war.
I am very dedicated and committed to winning the war against the rebels until recently when to my shock I found out that Senior Army Officers and Government officials were stealing public funds and looting government treasury and sending them to foreign countries. They have exploited the war situation to bring back the lawlessness and corruption of the Mobutu days.
After the late President Kabila was assainated in very questionable circumstances, I knew I was in danger of losing from both ends. Due to this development and as a way of survival, I have joined the train. I have in my possession funds amounting to US$15.5million. In the usual manner of conveying such funds, the money is in defaced form in a trunk box for security reasons. It is now in the safe custody of a security company. The company is however not aware of the real contentsof the trunk box.
My problem is that the financial Law of Nigeria does not give assylum seekers any financial rights. And because of the economic instability and religious crisis, I have decided that it will be unwise to attempt investing the funds here. The climate here is very unfavourable and as a result, I have not disclosed even to my hosts anything about this fund.
Now I am seeking foreign assistance to transfer the funds overseas. If you can assist, I am willing to give you 20% of the funds, that is US$3.1Million. You will understand that my entire life and future depend on this money and I shall be very grateful if you can assist me. The major thing I demand from you is the absolute assurance that the money will be safe and you will not sit on it when it is transferred to your account.
If you are ready to be of assistance, please contact me immediately by e-mail, so that I can furnish you with the modalities for the transaction and what is expected of you. I shall be most grateful if you maintain utmost confidentiality and keep this message entirely to yourself.

Best Regards,
MR. Edward Mambo.
234-80-330-46881
Email:edwardmambo@eudoramail.com

2002-08-27.html
STRICTLY CONFIDENTIAL

May I briefly introduce myself. I am DANGO SAKUSHI of the Democratic Republic of Congo (formerly Zaire) and the former aide de camp of the Late President Laurel Kabila of blessed memory.
I am presently living in Nigeria on asylum. In case you are wondering how I got your contact, I got your contact from a source at my country Embassy in Nigeria.
The events of the past two to three years in my country has been very unfortunate. The Late president Kabila successfully toppled the very corrupt government of the Late Dictator Mobutu Sese Seko and ruled justly and peacefully for a few months. Unfortunately, the Tutsi from Goma, the Northern part of the country rebelled against his government and since then my country has been at war.
I am very dedicated and committed to winning the war against the rebels until recently when to my shock I found out that Senior Army Officers and Government officials were stealing public funds and looting government treasury and sending them to foreign countries. They have exploited the war situation to bring back the lawlessness and corruption of the Mobutu days.
After the late President Kabila was assainated in very questionable circumstances, I knew I was in danger of losing from both ends. Due to this development and as a way of survival, I have joined the train. I have in my possession funds amounting to US$15.5million. In the usual manner of conveying such funds, the money is in defaced form in a trunk box for security reasons. It is now in the safe custody of a security company. The company is however not aware of the real contents of the trunk box.
My problem is that the financial Law of Nigeria does not give asylum seekers any financial rights. And because of the economic instability and religious crisis, I have decided that it will be unwise to attempt investing the funds here. The climate here is very unfavourable and as a result, I have not disclosed even to my hosts anything about this fund.
Now I am seeking foreign assistance to transfer the funds overseas. If you can assist, I am willing to give you 20% of the funds, that is US$3.1Million. You will understand that my entire life and future depend on this money and I shall be very grateful if you can assist me. The major thing I demand from you is the absolute assurance that the money will be safe and you will not sit on it when it is transfered to your account.
If you are ready to be of assistance, please contact me immediately by e-mail, so that I can furnish you with the modalities for the transaction and what is expected of you. I shall be most grateful if you maintain utmost confidentiality and keep this message entirely to yourself.
Please send your response to sakushi@lycos.com. My mobile line is 234-803-3297612 we are 1hr ahead GMT (London) time.

Best Regards,
DR. DANGO SAKUSHI.

Figure 2.4.1: Emails: 2001-10-11.html and 2002-08-27.html

The similarity between these two emails via the KF method is 1; the similarity between these two emails via the KFP method is 0.999992. Reading both emails shows that they are almost the same. For these two emails, both algorithms provided proper estimation of their similarity.

This does not hold in general, for the following example shows a "false positive" output by KF method. Consider the pair of emails: *2002-02-20a.html*, *2002-07-04b.html* (Figure 2.4.2).

Inspection of the documents clearly shows that they are written in very different styles. The similarities estimated by KF and KFP methods are 1 and 0.43177, respectively. It is evident that one is not able to distinguish these two emails by KF method, while the

2002-02-20a.html

Calvary Greetings.

I am a preacher with the Seed Harvest Ministry, and I will like to get advice from you. I believe the advice I need is secular to some extent, but do have the patience to understand my intention.
A few year's ago I was in Liberia where I had established a little congregation where I preached regularly, but the civil war escalated and the church was converted to a hospital of sought. On one faithfull day three Nigerian Soldiers came to me and left trunks of money with me and swore to come back for it.
When the war in Liberia intensified I deposited this box in a security vault in The Netherlands (Holland), I was escorted to Holland, by the ECOMOG soldiers who felt the content of the trunks were documents and never cared to check the trunks, I was air lifted to England with other refugees then, I stayed in England and when I was getting frustrated I returned to Africa only to find out that two of the three Soldiers died in Battle while the last was declared MIA (Missing in Action).
Now I have the fund, which the soldiers said was over thirty million dollars (US$30 million), but it lies in a security vault in a Security Company in Holland. I was contacted in a letter dated 12th of September by the security company asking me to indicate if I was in receipt of the letter as a sign that I am living and still subscribed for their service.
I am interested in using a small fraction of this money, much less than one percent for a re- organization of the work of God, but I do not need the rest and do not want to have any direct dealing with it, but I need someone who will be able to use the fund better maybe for charity or something universally profitable, I have thought of doing it myself but, my ministry is the apocalypse and I believe and preach the soon coming of the Lord which make me not indulgent in reliance on money or wealth in any form.
I also do not have the money that has accrued on the service and safe deposit of the company that holds the fund, I will be glad if you have any interest in the direction of maybe managing or transmitting the fund as you prefer in your capacity and understanding.
I will be glad to get a response from you ASAP.

Regards.
Rev. Sunday Nana

Email: 2002-02-20a.html

2002-07-04b.html

Pardon the abruptness and the liberty of this letter, It is due to its exigency.
I am a government functionary in my country. I Represent a caucus in need of a foreign partner to assist in the transfer and subsequent investment of a confidential sum in confidence.
We are ready to offer you a percentage for your assistance in the transfer and a negotiable fee if you can put together a portfolio for investment.
All we need is your trustworthiness and your capability to protect our interest.
Please note that we shall go through the legal procedures entailed in our laws and international laws in transferring the funds to you in trust. The source of the money is authoritative.
I wish that you would take the essence of this Letter in strict confidence.
You can reach me immediately for us to commence dialogue on the modi operandi entailed first, by email in earnest. Thank you for your courtesy.

Best regard,
Franklin Numa.

Email: 2002-07-04b.html

Figure 2.4.2:

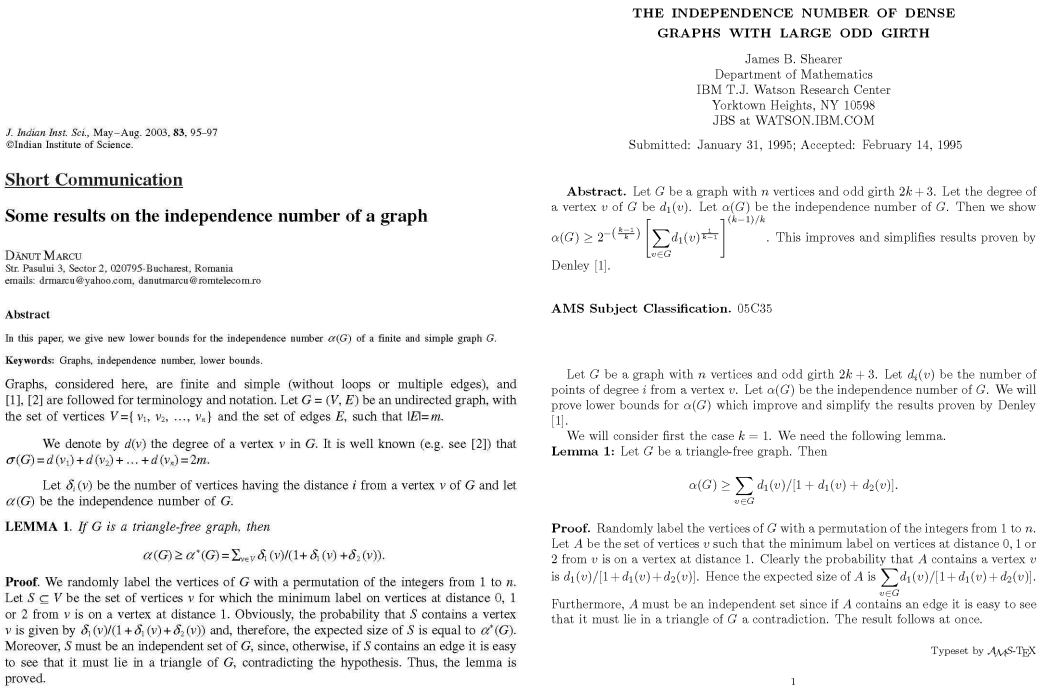
estimation of similarity by KFP method is much more reasonable.

2.4.2 Plagiarism Papers

Plagiarism in academic articles is a well-known issue. The widespread use of computers and the Internet has made it easier to plagiarize the work of others. Most cases of plagiarism are found in academia, where documents are typically scientific papers, essays or reports [25]. Our experiments show that the KFP method can be used to detect the plagiarism very efficiently.

In this case study our methodology involved the acquisition of a well-known plagiarized paper [26] (named *Paper-1A*) on the independence number of a graph and its corresponding original paper (named *Paper-1B*). In order to test whether our algorithm can detect the plagiarism, we randomly download a set of another 35 academic papers from the

internet (named *Paper-2, Paper-3, ... , Paper-36*), which are all related to the same subject, that is the independence numbers of graphs. Figure 2.4.3 is the first pair of papers: Paper-1A and Paper-1B .



J. Indian Inst. Sci., May–Aug. 2003, **83**, 95-97
©Indian Institute of Science.

Short Communication

Some results on the independence number of a graph

DĂNUŢ MARCU
Str. Pasului 3, Sector 2, 020795 Bucharest, Romania
emails: drmarcu@yahoo.com, danutmarcu@romtelecom.ro

Abstract

In this paper, we give new lower bounds for the independence number $\alpha(G)$ of a finite and simple graph G.

Keywords: Graphs, independence number, lower bounds.

Graphs, considered here, are finite and simple (without loops or multiple edges), and [1], [2] are followed for terminology and notation. Let $G = (V, E)$ be an undirected graph, with the set of vertices $V = \{v_1, v_2, \ldots, v_n\}$ and the set of edges E, such that $|E| = m$.

We denote by $d(v)$ the degree of a vertex v in G. It is well known (e.g. see [2]) that $\sigma(G) = d(v_1) + d(v_2) + \ldots + d(v_n) = 2m$.

Let $\delta_i(v)$ be the number of vertices having the distance i from a vertex v of G and let $\alpha(G)$ be the independence number of G.

LEMMA 1. *If G is a triangle-free graph, then*

$$\alpha(G) \geq \alpha^*(G) = \sum_{v \in V} \delta_1(v)/(1 + \delta_1(v) + \delta_2(v)).$$

Proof. We randomly label the vertices of G with a permutation of the integers from 1 to n. Let $S \subseteq V$ be the set of vertices v for which the minimum label on vertices at distance 0, 1 or 2 from v is on a vertex at distance 1. Obviously, the probability that S contains a vertex v is given by $\delta_1(v)/(1 + \delta_1(v) + \delta_2(v))$ and, therefore, the expected size of S is equal to $\alpha^*(G)$. Moreover, S must be an independent set of G, since, otherwise, if S contains an edge it is easy to see that it must lie in a triangle of G, contradicting the hypothesis. Thus, the lemma is proved.

THE INDEPENDENCE NUMBER OF DENSE GRAPHS WITH LARGE ODD GIRTH

James B. Shearer
Department of Mathematics
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598
JBS at WATSON.IBM.COM

Submitted: January 31, 1995; Accepted: February 14, 1995

Abstract. Let G be a graph with n vertices and odd girth $2k+3$. Let the degree of a vertex v of G be $d_1(v)$. Let $\alpha(G)$ be the independence number of G. Then we show $\alpha(G) \geq 2^{-\left(\frac{k-1}{k}\right)} \left[\sum_{v \in G} d_1(v)^{\frac{1}{k-1}}\right]^{(k-1)/k}$. This improves and simplifies results proven by Denley [1].

AMS Subject Classification. 05C35

Let G be a graph with n vertices and odd girth $2k+3$. Let $d_i(v)$ be the number of points of degree i from a vertex v. Let $\alpha(G)$ be the independence number of G. We will prove lower bounds for $\alpha(G)$ which improve and simplify the results proven by Denley [1].

We will consider first the case $k = 1$. We need the following lemma.

Lemma 1: Let G be a triangle-free graph. Then

$$\alpha(G) \geq \sum_{v \in G} d_1(v)/[1 + d_1(v) + d_2(v)].$$

Proof. Randomly label the vertices of G with a permutation of the integers from 1 to n. Let A be the set of vertices v such that the minimum label on vertices at distance 0, 1 or 2 from v is on a vertex at distance 1. Clearly the probability that A contains a vertex v is $d_1(v)/[1 + d_1(v) + d_2(v)]$. Hence the expected size of A is $\sum_{v \in G} d_1(v)/[1 + d_1(v) + d_2(v)]$.

Furthermore, A must be an independent set since if A contains an edge it is easy to see that it must lie in a triangle of G a contradiction. The result follows at once.

Typeset by $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX

1

Figure 2.4.3: Paper-1A and Paper-1B

All of the papers are obtained as *pdf* files. Due to the limitation of the technology, when we convert those pdf files into text files, mathematical formulas are not able to be converted in a proper way: the same formula from different pdf files may converted into very different sequences consisting of special symbols separating with various number of spaces. It will definitely introduce errors when calculating the distance between keywords. In order to eliminate the errors introduced when converting the pdf files into text files, we will use the number of alphabets between the keywords (instead of the number of words between keywords) as the distance between keywords.

The keywords set consists of 23 frequently used terminologies in graph theory. Table 2.4 and Table 2.5 indicate the significant difference in the applications of both methods: KF and KFP.

	KFP Method	KF Method
Similarity betweenPaper-1A and Paper-1B	0.778566	0.97074
Similarity between other pairs of papers	1. All less than 0.6. 2. Most of them are far less than 0.2.	6 pairs of papers have similarities greater than 0.97.

Table 2.4: Similarity Comparison 1

		Similarity By KFP Method	Similarity By KF Method
Paper-1A	Paper-1B	0.778566	0.97074
Paper-25	Paper-34	0.345626	0.994996
Paper-21	Paper-34	0.203773	0.985672
Paper-13	Paper-25	0.098588	0.980111
Paper-7	Paper-16	0.077647	0.973067
Paper-16	Paper-23	0.055026	0.971901

Table 2.5: Similarity Comparison 2

From Table 2.4, estimated by KFP method, the similarity between the Paper-1A (the plagiarism paper) and Paper-1B (the original paper) is 0.78, and the similarities between all other pairs of papers are less than 0.6, most of them are far less than 0.2. This strongly indicates that the KFP method works perfectly for the detection of a plagiarism paper.

However, if we use KF method, the similarity between the plagiarism paper and the original paper is 0.97074(see Table 2.5). And we also find other 6 pairs of papers have similarities greater than the similarity between the plagiarism paper and the original paper. For example, the similarity between Paper-25 and Paper-34 is above 0.99, (note that the similarity between these two papers by KFP method is 0.35). From Table 2.5, we can see that KFP method performs better than KF method.

2.5 Discussions

In this chapter, we introduced a weighted directed multigraph to model a text document. This method considers not only the keyword frequency information, but also the structure information in the form of the relations between keywords in documents. Through experiments performed on a set of emails and a set of research papers on graph theory, it is evident that the weighted directed multigraph model achieves significantly better than the commonly used frequency only model.

We performed experiments on two sets of documents. For the set of graph theory publications, publicly accessible knowledge about identified plagiarized papers provides us a meaningful "yardstick" for the measurement of the accuracy and effectiveness of our novel method. We may summarize our result with the following conclusion: *the KFP method is able to single out the plagiarized pair with the highest similarity which is much larger than any other pair of papers, while the KF method produces may results without any meaningful gap of similarity to distinguish positive and negative results.*

We also tried a weighted undirected multigraph model (i.e, neglect the direction from one keyword to the other keyword in the graph). Although it will lose some structure information of the document, the result is also very similar to what we described above. The advantage of undirected version is the significant reduction of the usage of memory space comparing with the weighted directed multigraph model.

These initial results indicated that the algorithm is much more effective at discriminating and clustering text documents and further improvement of accuracy and performance is expected. Specifically, it is anticipated that one can construct an ontological representation of the semantic information [32, 33, 34] to further enhance the KFP measure and that this information can then be used to set up the directed weighted multigraph. This will in turn allow us to use centrality guided clustering which will be presented in chapter 3 or QCM method to cluster all documents with even better precision.

Representing a document as a weighted directed multigraph model is the novel idea introduced in this chapter. This approach enables us to further distinguish documents from the SAME category into smaller groups base on writing style, or subcategory. We also believe this weighted directed multigraph model has a great potential to be applied to other data mining research in information related fields.

Chapter 3

A Centrality Guided Clustering (CGC)

3.1 Introduction

Clustering is the process of partitioning a set of data into meaningful subsets (called clusters or communities) so that every data in the same cluster are similar in some sense. It is a method of data exploration and a way of looking for patterns or structure in the data that are of interest. A general review of cluster analysis can be found in many references such as [40, 41, 42]. Clustering has wide applications in social science, biology, chemistry and information sciences. The system is usually modeled as a directed graph or undirected graph G. The set of vertices in G represents the objects under investigation, e.g., people or groups of people, molecular entities, computers, etc. Edges in G represent the relationship (such as friendship between people, co-authorship, protein-protein interaction) of the objects.

The commonly used clustering algorithms are hierarchical clustering algorithm and partitional clustering algorithms. Hierarchical clustering algorithms are either agglomerative or divisive. Agglomerative algorithms begin with each element as a separate cluster and merge them into successively larger clusters; divisive starts with one big cluster and splits are performed recursively as one moves down the hierarchy. Partitional algorithms typically determine all clusters at once, but can also be used as divisive algorithms in the hierarchical clustering. The K-means clustering algorithm is a typical partitional clustering technique. Given the number of cluster (say k), the procedure of K-means clustering is as following: (i) randomly generate k points as cluster centers, assign each point to

the nearest cluster center, (ii) recompute the new cluster centers. (iii) Repeat the two previous steps (i)(ii) until some convergence criterion is met. The main advantages of this algorithm are its simplicity and speed which allows it to run on large datasets. Its disadvantage is that it does not yield the same result with each run, since the resulting clusters depend on the initial random assignments.

In this chapter, we propose a novel idea of developing a new clustering algorithm: search of dense subgraphs leaded by centrality ranking of vertices. Traditional clustering methods, such as, K-means, usually choose clustering centers randomly. For the method being proposed here, we start clustering from the vertex with highest centrality score. By analyzing the communities of the social network data set, we will find that in each community there is usually some member (or leader) who plays an key role in that community. It is well known that centrality is an important concept [49] within social network analysis. High centrality scores identify members with the greatest structural importance in networks and these members would be expected to play a key role in the network. Based on the result of the centrality analysis of social network, we propose to begin clustering from the member with highest centrality score. That is, a community is formed starting from its "leader(s)", and a new "member" is added into an existing community based of its total relation with the community. We repeat the following steps: choose the vertex with highest centrality score which is not included in any existing community yet, we call this vertex "LEADER", and a new community is created with this vertex. We then repeatedly add one vertex to an existing community which satisfies the following criterion: the density of the newly extended community is above a given threshold.

The new clustering algorithm is described in Section 3.2. In Section 3.3, test results of the new algorithm on some popular bench-mark data sets are presented. Different centrality measurements will be discussed in section 3.5. Some discussions are presented in Section 3.4.

3.2 The Centrality Guided Clustering (CGC) Algorithm

In this section, we will present the the centrality guided clustering algorithm. We first introduce some notation which will be used in the algorithms.

Let C be a subgraph of G, we define the density of the subgraph C as

$$density(C) = \frac{2\sum_{e\in E(C)} w(e)}{|V(C)|(|V(C)|-1)}, \text{ if } |V(C)| > 1$$

The density of the subgraph C could be looked as the intra-cluster similarity. Good clustering should have high intra-cluster similarity and low inter-cluster similarity. i.e., we want $density(C)$ to be large.

Within graph theory and network analysis, centrality of a vertex measures the relative importance of a vertex within the graph (for example, how important a person is within a social network or how well-used a road is within an urban network). There are various measures of vertex centrality introduced in graph theory and data mining (see Section 3.5). We would expect that after clustering, each group has some center and the center has relative high centrality score. On the other hand, if we start from the vertex (called it a "LEADER") with high centrality score and add its neighbor to the group, we should expect those vertices with tight connection to the LEADER to be grouped together. The clustering result will have high intra-similarity and low inter-similarity. That is the motivation of the CGC algorithm. We denote the centrality score of the vertex v in the graph G as $score(v)$. In the CGC algorithm, the *subgraph centrality* [47] score is used to calculate the centrality score for each vertex in G. For any set S, the number of element in the set S is denoted as $|S|$.

For a vertex $v \notin V(C)$ and $v \in V(G)$, we define the contribution of v to C by

$$contribution(v,\ C) = \frac{\sum_{u\in V(C)} w(uv)}{|V(C)|}.$$

Given a subgraph C of G, we say a vertex $u \notin V(C)$ is a neighbor of C if there is a vertex $v \in C$ such that $uv \in E(G)$. We say u is a ***candidate neighbor*** of C if u satisfies the following three conditions: (a) u is a neighbor of the subgraph C, (b) there exists a vertex $v \in V(C)$ such that $w(u,v) \geq \alpha * max\{w(e)|e \in V(G)\}$ if $|V(C)| = 1$ or $contribution(u,\ C) > \beta * density(C)$ if $|V(C)| > 1$, (c) $score(u) < max\{score(v)|v \in C_i\}$. The candidate neighbor set is a neighbor that will be considered to be clustered to the current group C. Condition (b) is to control the decrease in the density of the current group after adding the candidate neighbor to that group. Condition (c) is saying that we only consider those vertices whose centrality score is lower than the centrality score of some vertex in C_i, i.e., if we find a vertex $v \in N(C_i)$ which has higher centrality score

than any vertices in C_i, this means the vertex v has already been clustered into a previous group so we will not group this vertex into the current group. α and β are used to control the clustering so that the density of the new subgraph will not decrease too much after we add a ***candidate neighbor*** to the subgraph C. We choose $\alpha = 0.8$, $\beta = 1 - \frac{1}{2*(|V(C)|+1)}$. In another paper [39], we proved that if $\alpha = 0.8$, $\beta = 1 - \frac{1}{2*(|V(C)|+1)}$, then the density of the new subgraph with a set of candidate neighbors added to the subgraph C will not decrease over $\frac{1}{3}$. We denote the set of all candidate neighbors of the subgraph C as $N(C)$.

The overall structure of the CGC algorithm is shown in Algorithm 3.2. The details of the GROUPING step is shown in Algorithm 3.2. For the GROUPING algorithm, we always start clustering a new community from the vertex with the highest centrality score which has not been clustered into any comunities yet, we call this vertex the center (or leader) of the new community. Denote this vertex as the current community C_i. Then we choose a vertex from the candidate neighbor set $N(C_i)$ which has the largest contribution to C_i comparing with all other vertices in $N(C_i)$.

The comunities after the GROUPing step may have some overlap elements. If the number of overlap elements in two comunities exceed some threshold, it is better to merge all vertices in the two groups into a new larger comunity. The MERGING algorithm (see Algorithm 3.2) describes the details about how to merge two comunities. In the MERGING algorithm, if the size of overlapping of two groups greater than half of the size of the small one of the two comunities, we merge the two comunities into one comunity.

After the MERGING step, we contract each comunity C_i as a new vertex v_i. If there is an edge between two comunities C_i and C_j, then there will be an edge v_iv_j in the contracted graph and the weight of the edge $w(v_i, v_j)$ is calculated in CONTRACTION Algorithm (see Algorithm 3.2), where $E(C_i, C_j)$ is the set of crossing edges which is defined as $E(C_i, C_j) = \{xy : x \in C_i, y \in C_j, x \neq y\}$.

Algorithm 3.1 CGC algorithm

Input: a weighted graph G

Output: clustering dendrogram of the graph G.

(Initialization) $l = 1,\ G_l = G$,

while $|V(G_l)| > 1$

(GROUPING) cluster the vertices in G_l into different comunities .

(MERGING) merge those comunities with large percentage of overlap.

(CONTRACTION) contract those vertices in the same comunity to a new vertex, calculate the edge weights in the contracted graph.

Denote the contracted graph as G_{l+1}, $l = l + 1$.

Algorithm 3.2 GROUPING Algorithm

Input: a weighted graph G_l

Output: each vertex is assigned to a comunity.

Calculate the centrality score of each vertex $v \in G_l$,

Order $v \in V(G_l)$ via their centrality scores, such that $Q = (v_1,\ v_2, ...,\ v_n)$ with $score(v_1) \geq score(v_2) \geq ... \geq score(v_n)$.

$i \leftarrow 0$

While $Q \neq \emptyset$

$i \leftarrow i + 1$

create a new comunity $C_i \leftarrow \{v_{i_1}\}$, where v_{i_1} is the first vertex in the vertex queue Q.

$New\_Q = Q - \{v_{i_1}\}$

While $|New\_Q| < |Q|$

$Q = New\_Q$

find the **candidate neighbor set** $N(C_i)$ of C_i.

calculate the contribution of all each vertex in $N(C_i)$.

Sort $v \in N(C_i)$ in decreasing order of contribution to C_i, i.e., $Q_N = (v_{n_1},\ v_{n_2}, ...,\ v_{n_k})$ where $v_{n_i} \in N(C_i)$ and $contribution(v_{n_1},\ C_i) \geq contribution(v_{n_2},\ C_i) \geq ... \geq contribution(v_{n_k},\ C_i),\ k = |N(C_i)|$.

If $N(C_i) = \emptyset$, break.

else

$C_i = C_i \cup \{v_{n_1}\}$

$New\_Q = Q - \{v_{n_1}\}$, $Q_N = Q_N - \{v_{n_1}\}$

Algorithm 3.3 MERGING Algorithm

Input: a weighted graph G_l which is already clustered into s comunities as in the GROUPING Algorithm.

Output: each vertex is assigned to a new comunity.

List all comunities of G_l as $L = \{C_1, C_2, ..., C_s\}$ such that
$|V(C_1)| \geq |V(C_2)| \geq ... \geq |V(C_s)|$

$h \leftarrow 2,\ j \leftarrow 1$
while $j < h$
If $|C_j \cap C_h| \geq \gamma * min(|C_j|, |C_h|)$,
$L = L \cup \{C_j \cup C_h\} - \{C_j\} - \{C_h\}$
$s \leftarrow s - 1,\ h \leftarrow max\{h - 2, 1\}$,
$h \leftarrow h + 1,\ j \leftarrow 1$
else
$j \leftarrow j + 1$

Algorithm 3.4 CONTRACTION Algorithm

Input: a weighted graph G_l which is already merged into s comunities as in the MERGING Algorithm,

Output: each vertex is assigned to a new comunity.

List all the merged comunities of G_l as $L = \{C_1, C_2, ..., C_s\}$
generate a vertex v_p for each comunity C_p.
for $i = 1$ to s
for $j = 1$ to s
$w(v_i, v_j) = \frac{\sum_{e \in E(C_i, C_j)} w(e)}{|V(C_i)| * |V(C_j)|}$
$w(v_j, v_i) = w(v_i, v_j)$

3.3 Applications of the Centrality Based Clustering Algorithm.

In this section, we present some applications of our method to some classical data sets and compare the results with the benchmarks.

3.3.1 Synthetic Network

In order to test the performance of the CGC algorithm, we generated a graph G_1 with 25 vertices as in Figure 3.3.1(a), and a graph G_2 with 81 vertices as in Figure 3.3.1(b), and we add two edges between G_1 and G_2 to generate the graph G as in Figure 3.3.2(a). This graph is an unweighted graph. Ideally, if we try some clustering algorithm on the network G, we would expect that the vertices in G_1 should be clustered in one group, and the vertices in G_2 should be in another group. Figure 3.3.2(b) shows the clustering result by CGC algorithm, the solid dots are clustered as one community and the squares are clustered as another community. The result of CGC algorithm on the graph G matches the expected result perfectly.

(a) G_1 (b) G_2

Figure 3.3.1: (a) is a graph with 25 vertices, (b) is a graph with 81 vertices

3.3.2 Zachary's Karate Club Dataset

Zachary's Karate Club Dataset is a typical dataset which is used to measure the clustering algorithm for social network. It is a social network of friendships between 34 members of a karate club at a US university in the 1970 [36]. Zachary recorded the interaction

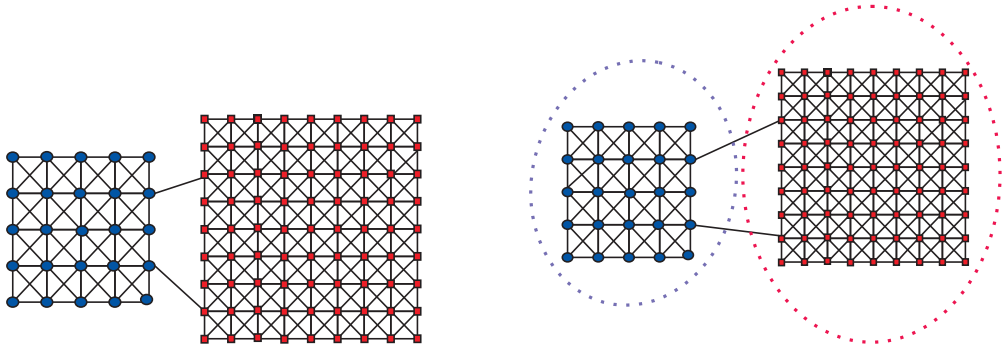
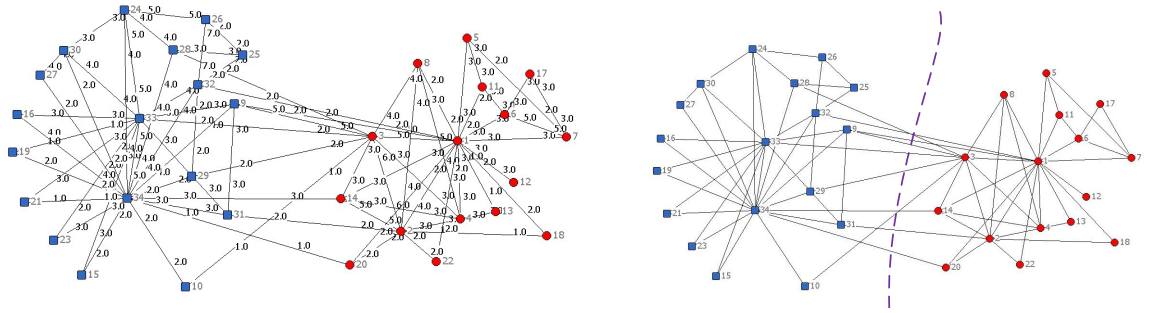


Figure 3.3.2: (a) is the graph G which has G_1 and G_2 as subgraphs. (b) is the clustering result of CGC algorithm, the solid dots are clustered as one community and the squares are clustered as another community.

of a Karate club in a university for two years. The graph of the karate club is in figure 3.3.3(a). The weight on the edge represents the interaction between different members. During the observation, the members in the Karate club encountered a disagreement that subsequently led to a formal separation of the club into two organizations. The result of CGC algorithm is shown in figure 3.3.3(b), the dashed line denotes the partition of the clustering result. The vertices on left side of the dotted curve belongs to one community and the vertices on the right hand side of the dashed curve belongs to another community. Figure 3.3.4 is the corresponding hierarchical tree based on CGC algorithm. The results are exactly same as the benchmark.

3.3.3 Dolphin Social Network

The dolphin social network dataset is another representative dataset to test the accuracy of clustering algorithms. It is an social network of frequent associations between dolphins in a community living off Doubtful Sound, New Zealand [35]. The social network of the dolphins are presented in the figure 3.3.5,there are 62 vertices and 159 edges in this network: vertices represent the bottlenose dolphins, edges between vertices represent associations between dolphin pairs occurring more often than expected by chance. During



(a) (b) clustering result

Figure 3.3.3: (a) is original graph of the Karate club. (b) the dotted line denotes the partition of the clustering result of CGC algorithm. The vertices on left side of the dashed curve belongs to one community and the vertices on the right hand side of the dashed curve belongs to another community.

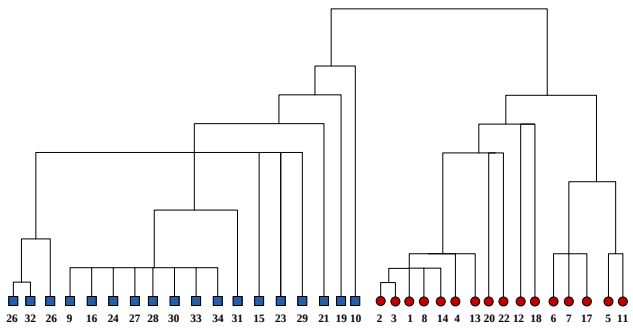


Figure 3.3.4: the hierarchical tree of the Karate Club dataset by the CGC algorithm

the course of the study, the dolphin group split into two smaller subgroups following the departure of a key member (represented as the triangle in the figure 3.3.5) of the population. The groundtruth subgroups are represented by the shapes of the vertices in the figure 3.3.5, the vertices represented as squares are in one group and the vertices represented as dots and triangle are in the other group. The dotted line denotes the division of the network into two equal-size groups found by the standard spectral partitioning method in which 11 out of 62 dolphins are misclassified. The solid curve represents the division found by the modularity-based method by Newman [38] in which 3 out of 62 dolphins are misclassified. We applied centrality guided clustering method to the dolphin social network, it divides the dolphins into two different groups giving the same result as the benchmark result. Figure 3.3.4 shows the corresponding hierarchical tree based on CGC algorithm.

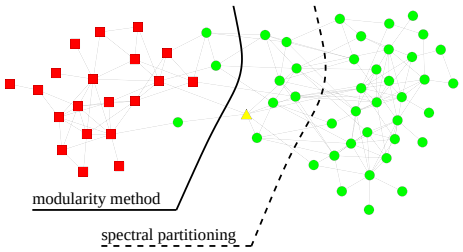


Figure 3.3.5: The social network of the dolphins. The dotted line denotes the division of the network into tow equal-size groups found by the standard spectral partitioning method, and the solid curve represents the division found by the modularity-based method by Newman [38].

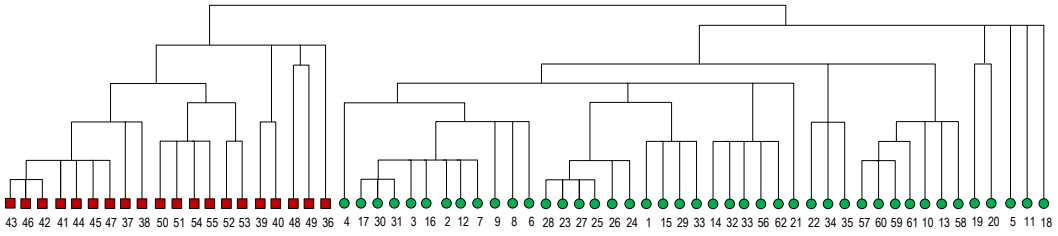


Figure 3.3.6: the hierarchical tree of the dolphin dataset based on CGC algorithm

3.3.4 The Fisher Iris Data

The iris data set published by Fisher (1936) has been widely used for examples in discriminant analysis and cluster analysis. The iris data set contains 150 iris specimens from three different species: setosa, versicolor, and virginica. For each species, 50 observations of the sepal length, sepal width, petal length and petal width of each iris are measured in millimeters.

This data set offers a challenge as two of the species are contiguous in the 4 dimensional space and are difficult to separate. The analysis of the Fisher Iris data using the K-means algorithm and Ward's algorithm are provided in SAS Institute [48]. A comparison of the

results using K-means, Ward's algorithm and our centrality guided clustering algorithm (CGC) are shown in table 3.1. The overall misclassification rate by K-means and Ward's algorithm are both 10.7%, The overall misclassification rate by CGC algorithm is only 8.7%.

	number of misclassification by different methods		
species	**K-means**	**Ward's method**	**CGC**
setosa	0	0	0
versicolor	2	1	4
virginica	14	15	9
total misclassification	16	16	13
error rate	10.7%	10.7%	8.7%

Table 3.1: comparison of the results of Iris data by different clustering methods

3.4 Discussions

In this chapter, we discussed the importance of the centrality score of vertices in a network and proposed a centrality guided clustering method. The CGC initiates the clustering processing at a vertex with high centrality score, which is a potential leader of a community. We have applied the CGC method to several benchmark data sets, the experimental results show that CGC algorithm outperforms existing clustering methods.

Different centrality measurements are described in section3.5. The degree criterion serves as a very local measurement for centrality, while betweenness centrality, closeness centrality search for global "leaders" of the entire networks. Since the goal of clustering is to identify local communities, instead of following global leaders, we suggest the use of subgraph centrality or **r** centrality as a subprogram in our proposed algorithm since it is an intermediate measurement of graph centrality. One research direction is to summarize different types of networks and figure out which centrality measurement works best for a given type of network system.

The CGC algorithm we developed is a hierarchical cluster algorithm. In the future, we plan to apply the centrality score guided idea to other clustering methods (such as K-means). We believe it will also produce promising clustering results.

3.5 Appendix: Measures of Centrality

Within graph theory and network analysis, the centrality of a vertex measures the relative importance of a vertex within the graph. There are various measures of the centrality of a vertex. Centrality measures, such as degree centrality, betweenness, closeness, eigenvector centrality and subgraph centrality, are among the most popular ones. Before we describe the centrality measures in detail, we first introduce some notation and terminology in graph theory which will be used in the this section and later sections.

Given an input dataset, we model the data set as a graph $G = (V, E, w)$. The vertices of G represent the elements of the dataset and the edges represent the relationships between different pairs of elements. $|V(G)|$ represents the number of vertices (or elements in the dataset). We use $w(u, v)$ or $w(e)$ to denote the weight of the edge e between two vertices u and v. If there is no edge between two vertices u and v, then we let $w(u, v) = 0$. If the graph is an unweighted graph, then

$$w(uv) = \begin{cases} 1 & \text{if } uv \in E(G) \\ 0 & otherwise \end{cases}$$

Degree Centrality

Given a graph G, the degree centrality for any $v \in G$ is defined as

$$c_D(v) = \frac{d(v)}{n-1}$$

where $d(v)$ is the degree of $v \in G$ and n is the total number of vertices in G. Degree centrality is a fundamental quantity describing the topology of scale-free networks. It can be interpreted as a measure of immediate influence, as opposed to longterm effect in the network [43].

Betweenness Centrality

Based on the assumption that transmission of information spreads along the shortest paths, betweenness centrality of the node for any $v \in G$ is defined as

$$c_B(v) = \frac{2}{(|V(G)|-1)(|V(G)|-2)} \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}},$$

where $s, v, t \in V(G)$, σ_{st} is the number of shortest paths from s to t, and $\sigma_{st}(v)$ is the number of shortest paths from s to t that pass through a vertex v. Betweenness centrality characterizes how influential a node is in communicating between node pairs [44].

Closeness Centrality

The closeness centrality is defined as the mean geodesic distance (i.e., the shortest path) between a vertex v and all other vertices reachable from it:

$$c_C(v) = \frac{|V(G)| - 1}{\sum_{u \in V \setminus v} dist(u, v)}$$

where $dist(u, v)$ is the length of shortest path from u to v.

Eigenvector Centrality

Eigenvector centrality is another popular measure of the importance of a node in a network. It assigns relative scores to all nodes in the network based on the principle that connections to high-scoring nodes contribute more to the score of the node in question than equal connections to low-scoring nodes. The eigenvector centrality score of the ith node in the network is defined as the ith component of the eigenvector corresponding to the greatest eigenvalue of the following characteristic equation $Ax = \lambda x$, where A is the adjacency matrix of the network, λ is the largest eigenvalue of A and x is the corresponding eigenvector. It simulates a mechanism in which each node affects all of its neighbors simultaneously [45]. Eigenvector centrality is better interpreted as a sort of extended degree centrality which is proportional to the sum of the centralities of the node' neighbors. Consequently, a node has high value of eigenvector centrality either if it is connected to many other nodes or if it is connected to others that themselves have high eigenvector centrality [46].

Subgraph Centrality

The subgraph centrality of any vertex i in a graph G is defined as the scaled sum of closed walks of different lengths in the network starting and ending at vertex i . As this sum includes both trivial and nontrivial closed walks we are considering all subgraphs,

i.e., acyclic and cyclic, respectively. The number of closed walks of length k starting and ending on vertex i in the network is given by the local spectral moments $\mu_k(i)$, which are simply defined as the ith diagonal entry of the kth power of the adjacency matrix A of G:

$$\mu_k(i) = (A^k)_{ii}$$

And the subgraph centrality of vertex i in the network is given by [47]:

$$c_S(i) = \sum_{k=0}^{\infty} \frac{\mu_k(i)}{k!}$$

Subgraph centrality measure characterizes the participation of each node in all subgraphs in a network. Smaller subgraphs are given more weight than larger ones, which makes this measure appropriate for characterizing network motifs.

Laplacian Centrality

Consider the undirected simple graph $G = (V, E, w)$. A simple graph is a graph without loop and parallel edges, so se have $w(v_i, v_i) = 0$ for any $v_i \in V(G)$. Since it is an undirected graph, we have $w(v_i, v_j) = w(v_j, v_i)$. For simplicity, we write $w(v_i, v_j)$ as $w_{i,j}$. Then the edge weight matrix of G can be written as

$$W(G) = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,n} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ w_{n,1} & w_{n,2} & \cdots & w_{n,n} \end{bmatrix} = \begin{bmatrix} 0 & w_{1,2} & \cdots & w_{1,n} \\ w_{1,2} & 0 & \cdots & w_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ w_{1,n} & w_{2,n} & \cdots & 0 \end{bmatrix}.$$

Define

$$X(G) = \begin{bmatrix} x_1 & 0 & \cdots & 0 \\ 0 & x_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & x_2 \end{bmatrix}$$

where $x_i = \sum_{j=1}^{n} w(v_i, v_j)$. Then the matrix $L(G) = X(G) - W(G)$ is called the Laplacian matrix of G. The Laplacian energy of G is defined as

$$E_L(G) = \sum_{i=1}^{n} x_i^2 + 2 \sum_{i<j} w_{i,j}^2.$$

The Laplacian centrality for any vertex $v \in V(G)$ is defined as

$$C_L(v) = \frac{E_L(G) - E_L(G - v)}{E_L(G - v)}$$

which is the relative change of the Laplacian energy caused by the deletion of the vertex from graph G.

Laplacian centrality unveils more structural information about connectivity and density around v. It is an intermediate measure between the global and local characterization of the importance (centrality) of a vertex.

Chapter 4

Postman Tour with Optimal Route-Pair Cost

4.1 Introduction

The Chinese postman tour problem is to find a shortest closed walk that visits every edge of a connected graph where a positive length is assigned to every edge. In this paper, we introduce a new optimization problem for finding an optimal closed walk where a route value matrix is defined at every vertex for every choice of consecutive edge pair along the walk. This problem is motivated by the DNA sequence assembling for solving the repeating subsequence problems.

We first introduce some mathematical notations and definitions which will be used in the later sections.

Given a simple directed graph G, for any $v \in V(G)$, denote the set of incoming neighbors of the vertex v as

$$N^-(v) := \{x : x \in G, xv \in E(G)\},$$

denote the set of outgoing neighbors of the vertex v as

$$N^+(v) := \{y : y \in G, vy \in E(G)\},$$

denote the set of incoming edges of the vertex v as

$$E^-(v) := \{xv : x \in G, xv \in E(G)\},$$

and denote the set of outgoing edges of the vertex v as

$$E^+(v) := \{vy : y \in G, vy \in E(G)\}.$$

The in-degree of v is denoted as $d^-(v)$ and the out-degree of v is denoted as $d^+(v)$.

Definition. Let G be a directed graph, G is ***Eulerian*** if and only if $d^-(v) = d^+(v)$ for any $v \in V(G)$.

Given a graph G, the weight $w : E(\mathrm{G}) \to Z$ is called ***Eulerian weight*** if the total weight of each edge-cut is even.

A directed graph is called ***strongly connected*** if for any $v \in V(G)$, there is a path from v to every other vertex.

A closed walk of a directed graph G is a ***postman tour*** if it passes through every edge at least once. We call a postman tour an ***optimal postman tour*** if each edge is passed by at least once and the total number of repeated edges is minimum. Given an optimal postman tour T_p of G, we denote $t(e)$ as the times of the edge e is visited in T_p. We say that $t(e) : E(G) \to Z$ is an ***optimal Eulerian weight*** corresponding to the optimal postman tour T_p.

Let $v \in V(G)$. For each $e_i \in E^-(v)$ and $e_j \in E^+(v)$, the pair $\{e_i, e_j\}$ is called a ***route-pair*** at v. For each $v \in V(G)$, define a **route-pair cost** $\phi_v : E^-(v) \times E^+(v) \mapsto R^+$. We call the value of $\phi_v(e_i, e_o)$ as the route value of the length two path $e_i v e_o$.

For each $v \in V(G)$, suppose $E^-(v) = \{e_1^-, e_2^-, ..., e_x^-\}$, $E^+(v) = \{e_1^+, e_2^+, ..., e_y^+\}$, we write a route-pair cost matrix at v as

$$R(v) = \begin{bmatrix} \phi_v(e_1^-, e_1^+) & ... & \phi_v(e_1^-, e_y^+) \\ \vdots & \vdots & \vdots \\ \phi_v(e_x^-, e_1^+) & ... & \phi_v(e_x^-, e_y^+) \end{bmatrix}$$

For a postman tour $W = v_1 e_1 v_2 e_2 \cdots e_i v_i e_{i+1} \cdots e_m v_1$, the ***total route-pair cost*** of W is defined as $\phi(W) = \sum_{i=1}^{m} \phi_{v_i}(e_i, e_{i+1}) + \phi_{v_1}(e_m, e_1)$. And we define the **Minimum route-pair cost (MRPC) Problem** as:

Given a strongly connected digraph G and a route-pair cost ϕ_v for every $v \in V(G)$,

find a postman tour $W = v_1e_1v_2e_2\cdots e_iv_ie_{i+1}\cdots e_mv_1$, with the total cost $\phi(W)$ as small as possible.

4.2 An Algorithm for Postman Tour with Minimum Route-pair Cost

In this section, we propose an algorithm to find a postman tour with minimum route-pair cost for the special case that G is Eulerian digraph with $d^-(v) = d^+(v) = 2$ for every vertex $v \in V(G)$.

If G is strongly connected Eulerian digraph with $d^-(v) = d^+(v) = 2$ for every vertex $v \in V(G)$, then the route matrix $R(v)$ for v could be written as

$$R(v) = \begin{bmatrix} \phi_v(e_1^-, e_1^+) & \phi_v(e_1^-, e_2^+) \\ \phi_v(e_2^-, e_1^+) & \phi_v(e_2^-, e_2^+) \end{bmatrix}$$

For this type of digraph G, we can always find a postman tour with minimum route-pair cost. The algorithm we proposed for this problem is called as MRPC algorithm, which is described in Algorithm 4.1. For any vertex $v \in V(G)$, if e_i is the incoming edge to v and it is followed by e_o in the postman tour, then we say that e_o is an ***subsequent edge*** of e_i, denote as $s(e_i) = e_o$.

Algorithm 4.1 the MRPC Algorithm

Input: A strongly connected digraph G with $d^-(v) = d^+(v) = 2$ and a route-pair cost function ϕ_v for $v \in V(G)$.

Output: a postman tour with minimum route-pair cost for G.

Step 1.

Substep 1.1 find subsequent edge for each incoming edge

denote the vertex set as $V = \{v_1, v_2, ..., v_n\}$,

For $i = 1$ to n, denote $E^-(v_i) = \{e_{i1}^-, e_{i2}^-\}$, $E^+(v_i) = \{e_{i1}^+, e_{i2}^+\}$.

$\triangle(v_i) = [\phi_{v_i}(e_{i1}^-, e_{i1}^+) + \phi_{v_i}(e_{i2}^-, e_{i2}^+))] - [\phi_{v_i}(e_{i2}^-, e_{i1}^+) + \phi_{v_i}(e_{i1}^-, e_{i2}^+)]$.

If $\triangle(v_i) < 0$,

then $s(e_{i1}^-) = e_{i1}^+$, $s(e_{i2}^-) = e_{i2}^+$;

else, $s(e_{i2}^-) = e_{i1}^+$, $s(e_{i1}^-) = e_{i2}^+$.

Substep 1.2 find all closed walks generated in Substep 1.1.

$\mathcal{W} = \emptyset$, $i = 0$

denote E as the edge set of graph G.

while ($E \neq \emptyset$)

$i \leftarrow i + 1$

choose the first edge e in E,

$W_i \leftarrow \{e\}$, $E \leftarrow E - \{e\}$

while $s(e) \notin W_i$

$W_i = W_i \cup \{s(e)\}$, $E \leftarrow E - \{s(e)\}$

$e = s(e)$

end

end

We get a series of edge-disjoint closed walk, say $\mathcal{W} = \{W_1, W_2, ..., W_k\}$.

Step 2.

If $k = 1$, then W_1 is the postman tour with minimum route-pair cost of G, DONE.

Else, we construct an auxiliary graph A for G as following:

(i) $V(A) = \{W_1, W_2, ..., W_k\}$,

(ii) $\forall\, v \in V(G)$, if v is in two closed walks, say W_i, W_j, then we add an edge to the auxiliary graph A. Label this edge as vW_iW_j and let $w(vW_iW_j) = |\triangle(v)|$.

Step 3. Find a minimum spanning tree T of the graph A by Kruskal's Algorithm.

Step 4. For each edge $vW_iW_j \in E(T)$, alternate the paths of W_i and W_j at the vertex v.

We claim that the MRPC ends up with one postman tour W. And the total route-pair cost of W is $\phi(W) = \sum_{i=1}^{k} \phi(W_i) + \sum_{e \in T} \triangle(e)$. We claim that the MRPC algorithm find one of the the postman tour with minimum rout-pair cost. And we will prove the correctness of the MRPC algorithm in the theorem 3.

Before we prove theorem 3, let us look at the example in Figure 4.2.1. Without defining any route value function, there are several different postman tours starting at the vertex v_1. Say for examples, $v_1av_2bv_3cv_4dv_5ev_1fv_4hv_2jv_5gv_3iv_1$ and $v_1fv_4dv_5gv_3cv_4hv_2bv_3iv_1av_2jv_5ev_1$. And there are other solutions. If we define route-pair cost at each vertex as following:

$$R(v_1) = \begin{bmatrix} \phi_{v_1}(e,\, a) & \phi_{v_1}(e,\, f) \\ \phi_{v_1}(i,\, a) & \phi_{v_1}(i,\, f) \end{bmatrix} = \begin{bmatrix} 10 & 1 \\ 4 & 3 \end{bmatrix}$$

$$R(v_2) = \begin{bmatrix} \phi_{v_2}(a,\, b) & \phi_{v_2}(a,\, j) \\ \phi_{v_2}(h,\, b) & \phi_{v_2}(h,\, j) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$R(v_3) = \begin{bmatrix} \phi_{v_3}(b,\, i) & \phi_{v_3}(b,\, c) \\ \phi_{v_3}(g,\, i) & \phi_{v_3}(g,\, c) \end{bmatrix} = \begin{bmatrix} 1 & 3 \\ 5 & 2 \end{bmatrix}$$

$$R(v_4) = \begin{bmatrix} \phi_{v_4}(c,\, d) & \phi_{v_4}(c,\, h) \\ \phi_{v_4}(f,\, d) & \phi_4(f,\, h) \end{bmatrix} = \begin{bmatrix} 15 & 2 \\ 1 & 4 \end{bmatrix}$$

$$R(v_5) = \begin{bmatrix} \phi_{v_5}(d,\, e) & \phi_{v_5}(d,\, g) \\ \phi_{v_5}(j,\, e) & \phi_{v_5}(j,\, g)) \end{bmatrix} = \begin{bmatrix} 1 & 12 \\ 8 & 6 \end{bmatrix}$$

Then by the Substep 1.1 in the MRPC algorithm we find that

at v_1: $s(e) = f$, $s(i) = a$,

at v_2: $s(a) = b$, $s(h) = j$,

at v_3: $s(b) = i$, $s(g) = c$,

at v_4: $s(f) = d$, $s(c) = h$,

at v_5: $s(d) = e$, $s(j) = g$,

And we find the walks

$$W_1 = efd,\ W_2 = iab,\ W_3 = hjgc$$

which could be rewritten as

$$W_1 = v_5ev_1fv_4dv_5,\ W_2 = v_3iv_1av_2bv_3,\ W_3 = v_4hv_2jv_5gv_3cv_4$$

Follow the step 2 in the MRPC algorithm, we get the auxiliary graph as in figure 4.2.2(a), there are one edge $v_1W_1W_2$ between W_1 and W_2, two edges $v_2W_2W_3$, $v_3W_2W_3$ between W_2 and W_3 and two two edges $v_4W_1W_3$, $v_5W_1W_3$ between W_1 and W_3. The weight of each edge is:

$w(v_1W_1W_2) = 8,\ w(v_2W_2W_3 = 2,\ w(v_3W_2W_3) = 5,\ w(v_4W_1W_3) = 16,\ w(v_5W_1W_3) = 13$.

By Kruskal's Algorithm, we find a minimum spanning tree as in figure 4.2.2(b) for the augment graph. If we switch the length-two paths through v_1 and v_2, we get exactly one walk $v_5ev_1av_2jv_5gv_3cv_4hv_2bv_3iv_1fv_4dv_5$ which traverses each edge exactly once. And this walk is equivalent to the path $v_1av_2jv_5gv_3cv_4hv_2bv_3iv_1fv_4dv_5ev_1$ which is starting and end at v_1. The total route-pair cost is 28.

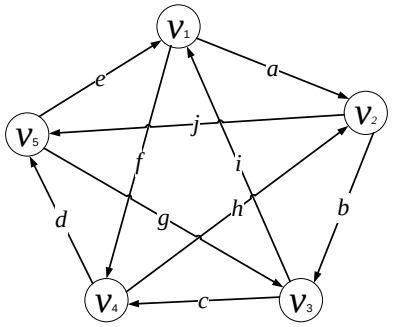
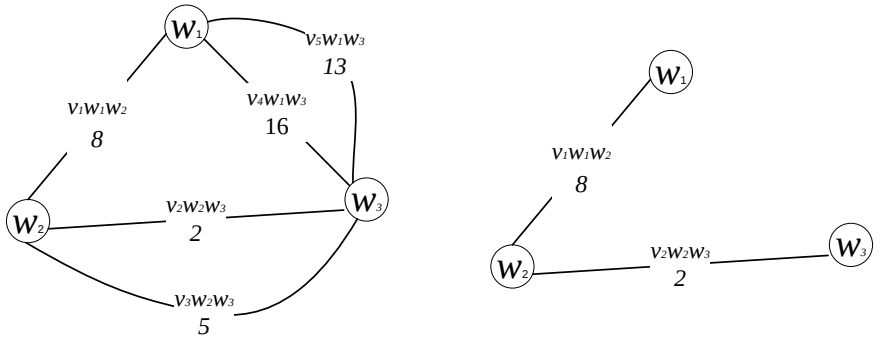


Figure 4.2.1: an example for the postman tour with minimum route-pair cost problem

Theorem 3. *Given a digraph G and a route-pair cost function ϕ_v for every $v \in V(G)$. If G is strongly connected and $d^-(v) = d^+(v) = 2$ for every $v \in V(G)$, then the **MRPC** Algorithm find one of the optimal postman tours with minimum route-pair cost.*

Proof. Claim 1. Step 1 in the **MRPC** algorithm produces a series of edge-disjoint closed walks that covers every edge precisely once.

By Substep 1.1 in the **MRPC** algorithm, every edge has only one subsequent edge and every edge is the subsequent edge of exactly one edge. So every edge is visited exactly once in the following steps in the **MRPC** algorithm.



(a) (b)

Figure 4.2.2: (a) is the augment graph of the figure 4.2.1, (b) is the minimum spanning tree of (a)

Since $d^-(v) = d^+(v) = 2$ for every $v \in V(G)$, each walk in $\mathcal{W} = \{W_1, W_2, ..., W_k\}$ at the end of step 1 in the **MRPC** algorithm is a closed walk. And for any two walks $W_i, W_j \in \mathcal{W}$, they must be edge-disjoint. Otherwise, some edge must be visited more than once.

Claim 2. Step 4 in the **MRPC** algorithm joins all the edge-disjoint walks.

For each edge $vW_iW_j \in E(T)$, we alternate the paths of W_i and W_j at the vertex v. Notice that $d^-(v) = d^+(v) = 2$, we have only two pairs of choices for the paths pass through v, either $\{e_1^-ve_1^+ \in W_i,\ \ e_2^-ve_2^+ \in W_j\}$ or $\{e_2^-ve_1^+ \in W_p,\ , e_1^-ve_2^+ \in W_q\}$. If $\{e_1^-ve_1^+ \in W_p,\ e_2^-ve_2^+ \in W_q\}$, then we let $s(e_2^-) = e_1^+$ and $s(e_1^-) = e_2^+$. Thus we alternate the paths through v to $e_2^-ve_1^+$ and $e_1^-ve_2^+$. As in the figure 4.2.3, $W_i = ve_1^+bp_iae_1^-v$ and $W_j = ve_2^+bp_jae_2^-v$ are edge-disjoint walks. After alternate the paths through v to $e_2^-ve_1^+$ and $e_1^-ve_2^+$, we get a new walk $ve_1^+p_iae_i^-ve_2^+p_je_2^1v$ which joins W_i and W_j. If $\{e_2^-ve_1^+ \in W_p$, $e_2^-ve_1^+ \in W_q\}$, then we let $s(e_1^-) = e_1^+$ and $s(e_2^-) = e_2^+$ and alternate the paths through v to $e_1^-ve_1^+$ and $e_2^-ve_2^+$.

Since T is a spanning tree of A, and each vertex of A is a walk in $\mathcal{W}$, the edges in T will join all walks in $\mathcal{W}$ to one walk.

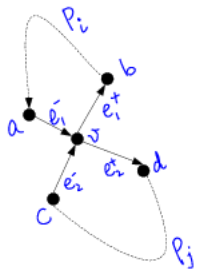


Figure 4.2.3: edge disjoint walk

By Claim 1 and Claim 2, we proved that the MRPC algorithm find a closed walk for the digraph G.

Suppose the closed walk (say $W = v_0e_1v_1e_2\cdots e_\mu v_\mu e_{\mu+1}\cdots e_m$) found by the MRPC Algorithm is not minimum, then there must be another closed walk, say $\widetilde{W}$, with $\phi(\tilde{W}) < \phi(W)$.

Let $\widetilde{W} = \tilde{v}_0\tilde{e}_1\tilde{v}_1\tilde{e}_2\cdots\tilde{e}_\mu\tilde{v}_\mu\tilde{e}_{\mu+1}\cdots\tilde{e}_m\tilde{v}_0$.

We first show that $\widetilde{W}$ could also be obtained from $\mathcal{W}$ by switching some length two paths at some vertices in G.

Construct an auxiliary graph $\tilde{A}$ for G as following:

(i) Let $V(\tilde{A}) = \{W_1, W_2, ..., W_k\}$ as described in Step 1 of the MRPC algorithm,

(ii) $\forall\, v \in V(G)$, if v is in two closed walks, say W_i, W_j, then we add an edge to the auxiliary graph A. We label this edge as vW_iW_j in order to distinct the parallel edges between the vertex W_i and the vertex W_j derived from different vertex in G, the weight of the edge vW_iW_j is defined as $w(vW_iW_j) = |\triangle(v)|$. Similarly, if v is in only one closed walk, say W_i, then add a loop to the auxiliary graph A. We label this loop as vW_iW_i in order to distinct the parallel loops in $\tilde{A}$ derived from different vertex $v \in G$, the weight of the loop vW_iW_i is $w(vW_iW_i) = |\triangle(v)|$.

(iii) Obviously, we can obtain the walk $\tilde{W}$ by connecting some walks in $\mathcal{W}$. [1] Suppose that $\tilde{T}$ is the subgraph of $\tilde{A}$ which is used to obtain the walk $\tilde{W}$, then $V(\tilde{T}) = V(\tilde{A})$, $E(\tilde{T})$ =\{the set of edges in $\tilde{A}$ that join all edge-disjoint walks in $\mathcal{W}$\}.

Claim 3: $\sum_{e\in T} w(e) \le \sum_{e\in\tilde{T}} w(e)$.

Since $\widetilde{W}$ is a closed walk, $\tilde{T}$ must be a connected spanning subgraph of the auxiliary

[1] (need to modify the description here)

graph $\tilde{A}$. Otherwise, some closed walk W_i, W_j will not be joined to the same component of the graph.

From the way we construct the auxiliary graphs A and $\tilde{A}$, it is easy to see that $A \subset \tilde{A}$ with $V(A) = V(\tilde{A})$. Since T is a minimum spanning tree of A, it is also a minimum spanning tree of $\tilde{A}$. So we must have $\sum_{e\in T} w(e) \leq \sum_{e\in \tilde{T}} w(e)$.

$\phi(\widetilde{W}) = \sum_{i=1}^{k} \phi(W_i) + \sum_{e\in \tilde{T}} w(e) \geq \sum_{i=1}^{k} \phi(W_i) + \sum_{e\in T} w(e) = \phi(W)$. Contradiction with $\phi(\widetilde{W}) < \phi(W)$. □

4.3 DNA Sequence Assembly

As we mentioned in section 4.1, the optimization problem defined in section 4.1 was motivated by DNA sequence assembling problem. Genome assembly problem is one of the fundamental problems within bioinformatics research since the Sanger sequencing method was introduced in 1977. A critical stage in genome sequencing is the assembly of shotgun reads. Genome is broken into small reads whose sequence is then determined. The problem of combining these reads to reconstruct the source genome is known as genome assembly.

There are two different types of genome assembly approaches. One is Comparative approach and the other is de novo approach. The comparative assembly use some available reference sequence as a guide during the assembly process. This approach is usually used when studying genomic variation within population. The de novo approach aimed at reconstructing genomes that are not similar to any organism previously sequenced. The main strategies for de novo assembly are overlap-layout-consensus (such as Celera Assembler and Arachne) and Eulerian path [54]. One of the problem related to de novo assembly is how to assemble those repeat segments. The overlap-layout-consensus such as Celera assembler masks repeats, generate a large set of contigs.

In order to resolve the repeat problem, some local optimal selections have been proposed in some pioneering works [51, 52, 53] The mate-pair information was used to deal with the repeat problems. Chaisson etc.[50] suggested that "when there are multiple paths in the repeat graph between a mate-pair, we may choose a path with maximum support from mate-pairs". They locally used the mate-pair information for resolving repeat problems. However, if a route is optimized at every vertex, the resulting assembly may not be a single contig. Inspired by their work, we propose to generate route value function by

using mate-pair information and find the postman tour with minimum route value in the DNA sequence graph. Instead of the local greedy approaches which may result in many contigs, our main contribution for the DNA assembly problem is to find a global optimal solution for resolving the collapsed repeats with only one contig.

For a set of strings $R = \{r_1, r_2, ..., r_n\}$, define S_k as the collection of all k-mer strings from R. Given a set of reads $R = \{r_1, r_2, ..., r_n\}$, the de Bruijn graph $G(V, E)$ is defined as follows: $V(G)$ is the set of all $(k-1)$-mer strings from R, if there is a k-mer in S_k that has the $(k-1)$-mer u as prefix and $(k-1)$-mer v as suffix, then $uv \in E(G)$. Each k-mer corresponding to an edge in G. An example of a de Bruijn graph for the sequence is shown in Figure 4.3.1(a). A vertex v is called a source if $d(v^-) = 0$. A vertex v is called a sink if $d(v^+) = 0$. A vertex is called nonbranching vertex if $d^+(v) = d^-(v) = 1$. Based on the k-mer de Bruijn graph, we can replace all paths containing nonbranching vertices by a single edge as in in Figure 4.3.1(b) and we call it condensed k-mer de Bruijn graph. We can simplify the condensed de Bruijn graph as Figure 4.3.1(c). In order to find the DNA sequence for the de Bruijn graph given in Figure 4.3.1, we need to find a walk that pass through every edge in Figure 4.3.1(c) at least once. Notice that the edge e_2 and e_4 have to be visited twice if we want to find a walk visit all edges. The possible walks are

$$v_1e_1v_2e_2v_3e_3v_2e_2v_3e_4v_4e_5v_3e_4e_6v_5$$

and

$$v_1e_1v_2e_2v_3e_4v_4e_5v_3e_3v_2e_2v_3e_4v_4e_6v_5$$

, the corresponding two possible DNA alignments are

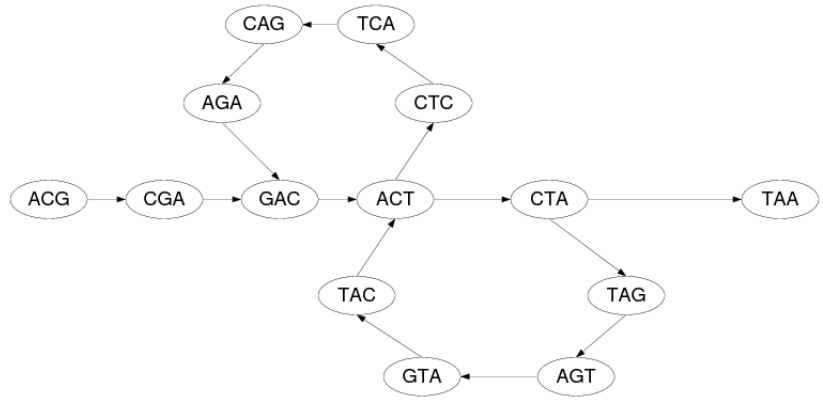
$$ACGACTCAGACTATACTAA$$

and

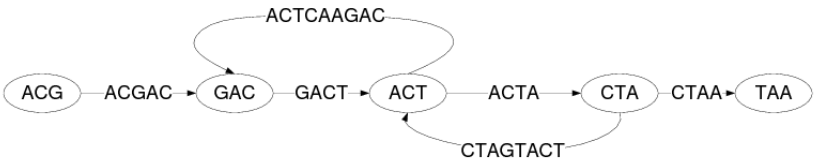
$$ACGACTAGTACTCAAGACTAA.$$

In order to determine which one of them are the correct alignment, we need to use some extra information. The mate-paired reads have been considered essential to de novo sequencing. Reads (denote as (r_1, r_2)) from opposite ends of the same clone insert are

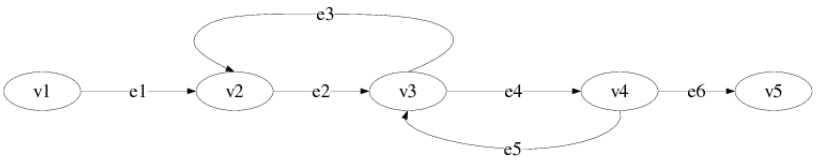
known as clone mates. The distance between the mate pair (r_1, r_2) is always given. After mapping every read to a path in the de Bruijn graph, we can map the mate pair (r_1, r_2) to some path in the de Bruijn graph that connects the positions of the mate pair (r_1, r_2) with path length approximately matches the clone length. In some case, this mapping may not be unique as in figure 4.3.2, the edge e_3 and e_5 have the same length.



(a)



(b)



(c)

Figure 4.3.1: (a) is a 4-mer de Bruijn Graph of the sequence, (b) is the condensed 4-mer de Bruijn Graph, (c) is a simplified graph of (b)

Based on the possible paths corresponding to each mate pairs, we can define the edge weight of the de Bruijn graph as following:

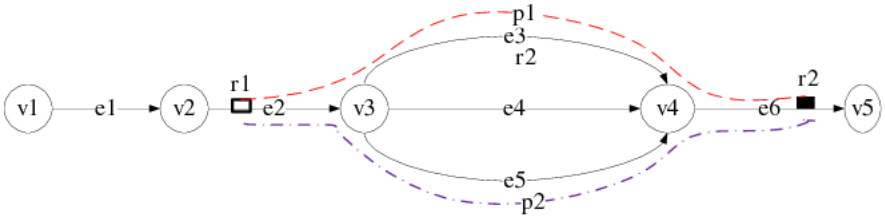


Figure 4.3.2: mate pairs which could be mapped to multiple paths in the de Bruijn graph

For any vertex v in the graph G, if $e_i \in E^-(v)$ and $e_j \in E^+(v)$, we let $\phi_v(e_i, e_j) = -$(number of mate pair paths that pass through $e_i v e_j$).

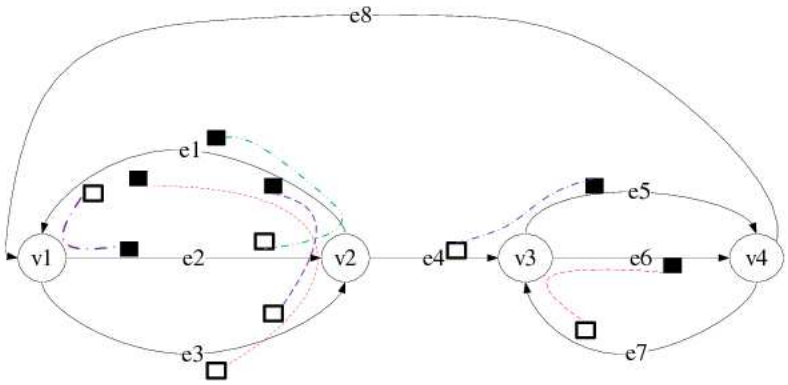


Figure 4.3.3: An example of a de Bruijn graph with mate pairs

If we use Figure 4.3.3 as an example, then the route matrix at v_2 is

$$R(v_2) = \begin{bmatrix} \phi_{v_2}(e_2, e_1) & \phi_{v_2}(e_2, e_4) \\ \phi_{v_2}(e_3, e_1) & \phi_{v_2}(e_3, e_4) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 2 & 0 \end{bmatrix}.$$

After we define a route matrix for each vertex $v \in G$, the DNA sequence assembly problem could be formulated as finding the postman tour with minimum route-pair cost problem. So at the vertex v_2, we will choose $e_3 v_2 e_1$ and $e_2 v_2 e_4$ as the two paths that pass through v_2 by the MRPC Algorithm in section **4.2.** The MRPC Algorithm will choose $e_4 v_3 e_5$ and $e_7 v_3 e_6$ as the two paths that pass through v_3. If v_1 is the starting vertex and

end vertex, then the final assembly by the MRPC Algorithm for the sequence in figure 4.3.3 will be $v_1e_3v_2e_2v_2e_4v_3e_5v_4e_7v_3e_6v_4v_1$.

Note that, the de Brujin graph may not be an eulerian digraph. If the de Brujin graph has only one source and one sink, then we can add an edge from the sink to the source. then we find the number of repeat of each edge by finding a minimal flow in a network with lower capacity bounds and get an Eulerian graph [51].

4.4 Different Optimization Problems with Route-Pair Cost

In section 4.2, we only discussed the solution for any connected Eulerian digraph with $d^-(v) = d^+(v) = 2$, $\forall\, v \in V(G)$. If there are some vertices with $d^-(v) > 2$ or $d^+(v) > 2$, then we conjecture that the problem is NP complete.

Further more, if the graph is not Eulerian, then some edges will be used multiple times. Which edge will be used in multiple times? For the optimal problem without route-pair cost matrix, it is simply the Chinese postman problem. However, with route-pair costs at vertices, it is a more complicated optimization problem. Here we propose some different optimization problems with route-pair cost for non-Eulerian digraph.

Given a strongly connected digraph G and a route-pair cost ϕ_v for every $v \in V(G)$. We propose the following problems:

- (I) If the optimal Eulerian weight of G is given as $t(e) : \ E(G) \to Z$, find an optimal tour $W = v_1e_1v_2e_2 \cdots e_iv_ie_{i+1} \cdots e_mv_1$ such that every edge e is visited $t(e)$ times and the total route cost $\phi(W)$ is minimum.

- (II) Under the restriction that $t(e) : \ E(G) \to Z$ is an optimal Eulerian weight of G , find an optimal tour $W = v_1e_1v_2e_2 \cdots e_iv_ie_{i+1} \cdots e_mv_1$ such that every edge e is visited $t(e)$ times and the total route cost $\phi(W)$ is minimum.

- (III) Find an optimal tour $W = v_1e_1v_2e_2 \cdots e_iv_ie_{i+1} \cdots e_mv_1$ such that every edge e is visited at least once and the total route cost $\phi(W)$ is minimum.

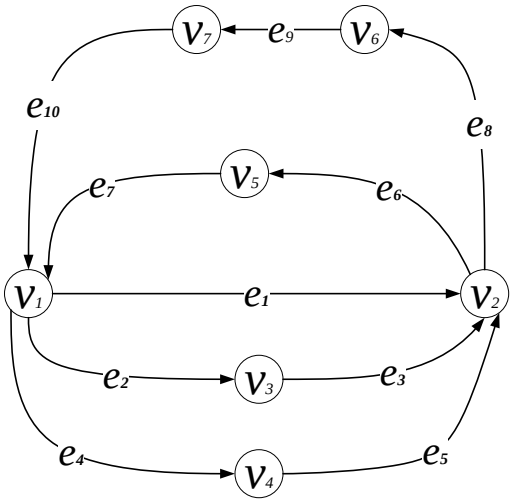


Figure 4.4.1: an example with different optimal solution to problem (I) and (II)

Notice that if a graph G is given, the total route costs of the optimal tours found in the problem (I), (II) and (III) may be different from each other. And the optimal tour of G found in the problem (III) may not be an optimal postman tour of G. The following (Figure 4.4.1) is an example with respect to it.

Suppose the route costs are $\phi_{v_2(e_1,e_6)} = 100$, $\phi_{v_2(e_1,e_8)} = 600$, $\phi_{v_2(e_3,e_6)} = 200$, $\phi_{v_2(e_3,e_8)} = 3$, $\phi_{v_2(e_5,e_6)} = 500$, $\phi_{v_2(e_5,e_8)} = 4$, , all other route costs are 0. The tour is starting from v_1 and end at v_1. Then the one of the optimal Eulerian weight will be:

$$\begin{cases} t(e_i) = 2 & \text{if } i = 6, 7 \\ t(e_i) = 1 & \text{otherwise} \end{cases} \quad (*)$$

One of the optimal solutions for problem (I) with the given optimal Eulerian weight in (*) is $v_1e_1v_2e_6v_5e_7v_1e_2v_3e_3v_2e_6v_5e_7v_1e_4v_4e_5v_2e_8v_6e_9v_7e_{10}v_1$. The total route weight is $100 + 200 + 4 = 304$. In this tour, only e_6 and e_7 are visited twice, all other edges are visited once.

If we consider problem (III) for the same graph, we find one of optimal tours is $v_1e_1v_2e_6v_5e_7v_1e_2v_3e_3v_2e_8v_6e_9v_7e_{10}v_1e_4v_4e_5v_2e_8v_6e_9v_7e_{10}v_1$. The total route weight is $100 + 3 + 4 = 107$. In this tour, e_8, e_9 and e_{10} are visited twice, all other edges are visited once. Notice that this tour is not an optimal postman tour without considering the route-pair cost.

By this example, we want to show that under different conditions (whether we want an optimal postman tour), we get different solutions for the same route-pair cost graph. If we want to get an optimal tour with the minimum route-pair cost under the precondition that the tour should be an optimal postman tour, then it is problem (III). In that case, we usually need to find the optimal eulerian edge weight. Section 4.5 will present how to find the optimal Chinese postman tour for an arbitrary unweighted digraph.

4.5 Appendix: Chinese Postman Tour

In this section, we discuss how to find the optimal solution and repeat times of each edge for the Chinese postman tour problem.

Lemma 4. *A digraph G contains a postman walk if and only if G is strongly connected.*

Proof. " $\Rightarrow$ " Suppose G is not strongly connected, then there exists some $v, u \in V(G)$, such that there is no path from v to u. If there exists a postman walk in the digraph G, we can write the walk as $W_1 = v_0p_1vp_2up_3v_0$ or $W_2 = v_0p_1upvp_3v_0$. If W_1 is the postman walk, then vp_2u contains a path from v to u. If W_2 is the postman walk, then $vp_3v_0p_1u$ contains a path from v to u. Contradiction with G is not strongly connected.

" $\Leftarrow$ " If G is strongly connected, choose a longest closed direct walk, say $C = v_1e_1v_2...e_nv_1$, which contains as many edges as possible. If every edge is in C, then C is a postman walk of G. If there is an edge, say $e = uw$ is not included in the walk C, then there must be a path p_u from v_1 to u and a path p_w from w to v_1, then $\widetilde{C} = v_1e_1v_2...e_np_up_w$ is a walk longer than the walk C. Contradict with C is the longest walk. □

If G is strongly connected but G is not an Eulerian graph, the postman walk must visit some edge of G more than once.

For any $v \in V(G)$, let $\delta(v) = d^-(v) - d^+(v)$, let $S = \{v|v \in V(G) : \delta(v) > 0\}$ and $T = \{v|v \in V(G) : \delta(v) < 0\}$. We then create a cost-capacity network $N = (G_A, b, c)$, where G_A is an auxiliary graph for the graph G. It is created as following:

(1) G_A is obtained from G by adding two extra vertices s and t.

(2) for each $v \in S$, add an edge from s to v into $E(G_A)$, $b(sv) = 0,\ c(sv) = \delta(v)$

(3) for each $v \in T$, add an edge from v to t into $E(G_A)$, $b(sv) = 0,\ c(sv) = -\delta(v)$

(4) for each $e \in G$, add an edge e into $E(G_A)$, $b(e) = 1,\ c(e) = \infty$

In order to find the minimum multiplicity of each edge e in G, it is equivalent to find the solution of the minimum-cost maximum-flow problem in the network $N = (G_A, b, c)$. And the minimum-cost maximum-flow problem can be solved by applying Klein's algorithm [55].

Bibliography

[1] N.E. Huang et al., The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis, Proc. Roy. Soc. London A 454 (1998) 903-995.

[2] Norden E. Huang. Introduction to the Hilbert-Huang Transform and Its Related Mathematical Problems. In Huang, N.E., Shen, S.S.P., The Hilbert-Huang Transform and Its Applications, vol. 5. World Scientific Publishing, Hackensack, NJ.

[3] Qiuhui Chen, Norden Huang, Sherman Riemenschneider and Yuesheng Xu. A B-spline approach for empirical mode decompositions. Adv. Comput. Math. 24 (2006), no. 1-4, 171–195.

[4] S.Riemenschneider, B.Liu, Y.Xu, and N.E.Huang, B-spline Based Empirical Mode Decomposition, In Hilbert-Huang Transform and its Applications Interdisciplinary Mathematical Sciences - Vol. 5, World Scientific, ISBN 981-256-376-8, Hong-Kong, 2005, pp 27-56.

[5] G. Rilling, P. Flandrin, and P. Gonçalves, On empirical mode decomposition and its algorithms, in Proceedings of IEEE-EURASIP Workshop on Nonlinear Signal and Image Processing (NSIP '03), Grado, Italy, June 2003

[6] www.ens-lyon.fr/~flandrin/software.html

[7] J.M. Ponte andW. B. Croft. A language modeling approach to information retrieval. In Research and Development in Information Retrieval, pages 275-281, 1998.

[8] Ruiz, M. E. and Srinivasan, P. Hierarchical text categorization using neural networks. Information Retrieval, 5(1), pp. 87-118, 2002.

[9] Support Vector Machine Active Learning with Applications to Text Classification. Simon Tong, Daphne Koller. Journal of Machine Learning Research, Volume 2, pages 45-66, 2001.

[10] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In Proceedings of the 14th International Conference on Machine Learning, Nashville, US, 1997.

[11] H. Schutze, D. A. Hull, and J. O. Pedersen. A comparison of classifiers and document representations for the routing problem. In Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval, Seattle, Washington, 1995.

[12] M. Lan, C. Tan, H. Low, and S. Sungy. A comprehensive comparative study on term weighting schemes for text categorization with support vector machines. In Proceedings of the 14th international conference on World Wide Web, pages 1032-1033, 2005.

[13] R. Robertson and K. Sparck-Jones. Simple, proven approaches to text retrieval. Technical report, 1997.

[14] Olivier de Vel , A.Anderson, M..Corney and G.Mohay. Mining E-Mail Content for Author Identification Forensics. SIGMOD Record, Vol. 30, No. 4, pages 55-64, 2001.

[15] C. Apte, F. Damerau, and S. Weiss. Text mining with decision rules and decision trees". In Workshop on Learning from text and the Web, Conference on Automated Learning and Discovery, 1998.

[16] H. Ng, W. Goh, and K. Low. Feature selection, perceptron learning, and a usability case study for text categorization. In Proc. 20th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR97), pages 67-73, 1997.

[17] T. Mitchell. Machine Learning. McGraw-Hill, New York, 1997.

[18] Y. Yang and X. Liu. A re-examination of text categorisation methods. In Proc. 22nd Int. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR99), pages 67-73, 1999.

[19] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In Proc. European Conf. Machine Learning (ECML'98), pages 137-142, 1998.

[20] P. Jackson and I. Moulinier. Natural Language Processing for Online Applications: Text Retrieval, Extraction, and Categorization. John Benjamins Publishing Co, 2002.

[21] D. D. Lewis. Naive (Bayes) at forty: The independence assumption in information retrieval. In Proceedings of the 10th European Conference on Machine Learning (ECML), pages 4–15, Chemnitz, Germany, 1998.

[22] F. Sebastiani. Machine learning in automated text categorization. ACM Computing Surveys, 34(1) pages 1–47, 2002.

[23] Samer Hassan, Rada Mihalcea and Carmen Banea, Random-Walk Term Weighting for Improved Text Classification. In Proceedings of the IEEE International Conference on Semantic Computing (ICSC 2007), Irvine, CA, September 2007.

[24] Nigerian Fraud Email Gallery. http://potifos.com/fraud/.

[25] http://en.wikipedia.org/wiki/Plagiarism.

[26] http://en.wikipedia.org/wiki/D%C4%83nu%C5%A3_Marcu.

[27] W. Hardle and L. Simar. Applied Multivariate Statistical Analysis. Berlin, Springer, 2003.

[28] P. Hansen and B. Jaumard. Cluster analysis and mathematical programming. Mathematical Programming, pages 191-215, 1997.

[29] G. W. Milligan. Cluster analysis. in Encyclopedia of Statistical Sciences (S. Kotz (Ed.), New York, NY: Wiley, pages 120-125, 1998.

[30] Y. Xu, V. Olman and D. Xu. Clustering gene expression data using graph-theoretic approach: an application of minimum spanning trees. Bioinformatics, 18 (2002), pages 536-545.

[31] Yongbin Ou and Cun-Quan Zhang. A new multimembership clustering method. Journal of Industrial and Management Optimization, Vol. 3, No. 4 (2007), pages 619-624.

[32] Landauer TK, Foltz P, Laham D. An introduction to latent semantic analysis. Discourse Processes 25(1998), pages 259-284.

[33] B. Rosario. Latent Semantic Indexing: An overview. INFOSYS 240 (Spring 2000).

[34] Yves Bestgen. Improving Text Segmentation Using Latent Semantic Analysis: A Reanalysis of Choi, Wiemer-Hastings, and Moore. Computational Linguistics, Volume 32 , Issue 3(2006), pages 455 - 455.

[35] D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson, Behavioral Ecology and Sociobiology 54, 396-405 (2003).

[36] W. W. Zachary, An information flow model for conflict and fission in small groups, Journal of Anthropological Research 33, 452-473 (1977).

[37] V. M. Janik and P. J. B. Slater, Context-specific use suggests that bottlenose dolphin signature whistles are cohesion calls, Animal Behaviour Volume 56, Issue 4, October 1998, Pages 829-838.

[38] Newman, M. E. J.,2006. Finding community structure in networks using the eigenvectors of matrices, Phys. Rev. E, 2006, 74, 036104.

[39] Yongbin Ou and Cunquan Zhang, "A new multi-membership clustering method," Journal of Industrial and Management Optimization, vol 3(4), pp. 619–624, 2007.

[40] G. W. Milligan, Cluster analysis, in "Encyclopedia of Statistical Sciences" (S. Kotz (Ed.), New York, NY: Wiley, (1998), 120-125.

[41] W. H¨ardle, and L. Simar, "Applied Multivariate Statistical Analysis," Berlin, Springer, 2003.

[42] P. Hansen, and B. Jaumard, Cluster analysis and mathematical programming, Mathematical Programming, 79 (1997), 191-215.

[43] Freeman, L.C., 1979.Centrality in social networks conceptual clarification. Social Network 1, 215-239.

[44] Freeman, L.C., 1977. A Set of Measures of Centrality Based Upon Betweeness. Sociometry 40, 35-41.

[45] P. Bonacich, Am. J. Sociol. 92, 1170 (1987).

[46] M. E. J. Newman, Phys. Rev. E 70, 056131 (2004).

[47] Estrada, E., Rodriguez-Velazquez, J.A. (2005) Subgraph centrality in complex networks. Physical Review E, 71(5): 056103

[48] SAS Institute (1999) SAS/STAT User's Guide. Cary: SAS Institute.

[49] Wasserman, S. & Faust, K. (1994), Social Network Analysis: Methods and Applications, Cambridge University Press.

[50] Chaisson MJ, Brinza D, Pevzner PA. 2009. de novo fragment assembly with short mate-paired reads: Does the read length matter? Genome Res. 19:336-46.

[51] Pevzner, P.A. and Tang, H. 2001. Fragment assembly with double-barreled data. Bioinformatics 17: S225–S233.

[52] Pevzner, P.A., Tang, H., and Waterman, M.S. 2001. A Eulerian path approach to DNA fragment assembly. Proc. Natl. Acad. Sci. 98: 9748– 9753.

[53] Pevzner, P.A., Tang, H., and Tesler, G. 2004. de novo repeat classification and fragment assembly. Genome Res. 14: 1786–1796.

[54] Mihai Pop, Genome assembly reborn: recent computational challenges. Brief Bioinform. 2009 Jul;10(4):354-66. Epub 2009 May 29.

[55] Bernhard Korte and Jens Vygen, Combinatorial Optimization: Theory and Algorithms (Algorithms and Combinatorics, 21), 4th ed., 2008, Springer.