FLORIDA STATE UNIVERSITY

COLLEGE OF ARTS AND SCIENCES

SOLVING CLUSTERING PROBLEMS USING RIEMANNIAN OPTIMIZATION

By

MENG WEI

A Dissertation submitted to the
Department of Mathematics
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

2023

Meng Wei defended this dissertation on July 5, 2023.
The members of the supervisory committee were:

Kyle A. Gallivan

Professor Co-Directing Dissertation

Wen Huang

Professor Co-Directing Dissertation

Gordon Erlebacher

University Representative

Giray Okten

Committee Member

Mark Sussman

Committee Member

The Graduate School has verified and approved the above-named committee members, and certifies that the dissertation has been approved in accordance with university requirements.

I dedicate this dissertation work to my parents and family for their constant encouragement and support during the challenges of graduate school and life.

I also dedicate this dissertation to my academic mentors, Kyle Gallivan, Wen Huang and Paul Van Dooren. They not only enlightened me with the academic knowledge but also gave me invaluable advice whenever I needed it the most.

ACKNOWLEDGMENTS

I would like to acknowledge all the following people who helped me during my PhD journey.

First and foremost, I would like to express the deepest gratitude to my advisors, Professor Kyle Gallivan, Professor Wen Huang and Professor Paul Van Dooren, for their invaluable advice, continuous support, and patience during my PhD study. They are wonderful mentors and researchers. Their immense knowledge and experience have encouraged me in all the time of my academic research and daily life. Their guidance has improved my research skills and independent thinking skills and prepared me for future challenges.

Secondly, I would like to thank my committee members, Professor Gordon Erlebacher, Professor Giray Okten, and Professor Mark Sussman, for their time and commitment. Their constructive comments and remarks have greatly improved the quality of my dissertation.

During my PhD study, I was funded as a teaching assistant in our department of mathematics. I would like to express my gratitude for the administrative and financial support of the department. In particular, I acknowledge the efficient assistance of Penelope Kirby and Elizabeth Scott. I also appreciate the financial support from DBI Grant 1934157.

I would also like to thank my friends, my fellow PhD students and office mates for their help and support during the process of pursuing my PhD. Thank you Xin Shan, Yiran Chen, Xiaoyu Wang, Sihui Liu,Yang Liu, Xin Li, Shuguang Zhang, Zhifeng Deng, Hua Huang, Hui Sun, Yijia Zhou, Jamie Fox, Suyog Shelar ... I simply can not name them all but none will be forgotten.

Last but definitely not least, I would like to give my special thanks to my family and especially my parents and my sister for their constant encouragements, patience and support to make this dissertation possible. To my life partner, Peng Diao and my daughter Selina: your love and understanding helped me to reach my goals. Without you believing in me, I would have never been able to achieve my dissertation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

ABSTRACT

This dissertation considers the optimization problems that are in the form of $\min_{X \in \mathcal{F}_v} f(x) + \lambda \|X\|_1$, where f is smooth, $\mathcal{F}_v = \{X \in \mathbb{R}^{n \times q} : X^T X = I_q, v \in \text{span}(X)\}$, and v is a given positive vector. Clustering analysis is a fundamental machine learning problem with broad-ranging applications. Its aim is to group unlabelled data objects into clusters according to a certain similarity measure such that objects within each cluster are more similar to each other than to objects of another cluster. Several clustering problems can be formulated as such optimization problems, such as community detection, k-means clustering, discriminative k-means clustering, and normalized cuts. In this dissertation, three Riemannian optimization approaches for the clustering problems in terms of assignment matrix are proposed, which are ARPPG, AManPG, and Inexact_AManPG. It is proven that the domain $\mathcal{F}_v$ forms a compact embedded submanifold of $\mathbb{R}^{n \times q}$ and optimization-related tools are derived. Global convergence analysis for Inexact_AManPG is then established. Numerical experiments on clustering problems including community detection, k-means, discriminative k-means, and normalized cut for image segmentation are used to demonstrate the performance of the proposed optimization approaches. ARPPG, AManPG, and Inexact_AManPG algorithms as with many other projection and dimension reduction methods, can have difficulties with time and space complexity when the number of clusters is not reliably known or too large. To address this limitation, we also propose a recursive version of Inexact_AManPG algorithm.

CHAPTER 1

INTRODUCTION

1.1 Motivation and Problem

Clustering analysis is a fundamental machine learning problem with broad-ranging applications. Its aim is to group unlabelled data objects into clusters according to a certain similarity measure such that objects within each cluster are more similar to each other than otherwise. The clustering algorithms vary based on the structure of the objects and the measurement of the similarity. For example, if the objects are located in a metric space and the similarity is measured by distance, then it is the standard clustering problem and many algorithms have been proposed to solve this problem, e.g., k-means clustering. k-means [70, 39, 47] is one of the most popular clustering methods. The basic idea of k-means algorithm is: given an initial but not optimal clustering, relocate each point to its new nearest center, update the clustering centers by calculating the mean of the member points, and repeat the relocating-and-updating process until convergence convergence criteria are satisfied. Recent research has improved the algorithm in many ways. One drawback to the k-means algorithm is that it is sensitive to the initialization of the centroids. To overcome this drawback, the k-means++ algorithm [112] is the standard initialization algorithm for the k-means algorithm. Another drawback to k-means is that it cannot separate clusters that are non-linearly separable in the input space. Two approaches have emerged to overcome this problem. One is applying kernel methods to k-means [101, 31, 37], which first embed the data into a higher dimensional feature space using a nonlinear function and then partitioning the points by linear separators in the new space. The other approach is spectral clustering algorithms [103, 86, 59], which use the eigenvectors of an affinity matrix to obtain a clustering of the data. A popular objective function used in spectral clustering is to minimize the normalized cut [103].

Boutsidis et.al. in [15] formally defined the k-means clustering problem using the cluster indicator matrix and then presented the first provably accurate feature selection algorithm for k-means clustering. The k-means algorithm can be rewritten as an alternating minimization algorithm [15]

for solving the optimization problem

$$\min_{X \in \mathcal{A}_{n,q}} ||A - XX^T A||_F^2, \tag{1.1}$$

where A is a data matrix where each data point in $\mathbb{R}^d$ corresponds to a row of A, and $\mathcal{A}_{n,q} = \{X \in \mathbb{R}^{n \times q} : X^T X = I_q, X \geq 0, \mathbf{1}_n \in \text{span}(X)\}$, $X \geq 0$ denotes that all entries of X are nonnegative, $\mathbf{1}_n$ denotes a vector with length n and all entries being 1, and $\text{span}(X)$ denotes the space spanned by column vectors of X.

When the data sets are in a very high dimensional space, a dimensionality reduction technique, Linear Discriminant Analysis (LDA) can be combined with the k-means clustering, e.g., Linear Discriminant Analysis plus k-means is presented in [33]. It is shown in [121] to be equivalent to a kernel k-means. The resulting optimization problem is then given by

$$\min_{X \in \mathcal{A}_{n,q}, \lambda_{regu} > 0} \text{trace}(X^T (I_n + \frac{1}{\lambda_{regu}} A^T A)^{-1} X) + \log\det(I_n + \frac{1}{\lambda_{regu}} A^T A), \tag{1.2}$$

where $\lambda_{regu} > 0$ is the regularization parameter. This problem is solved by alternating between the computation of X for a given λ_{regu} and the computation of λ_{regu} for a given X.

If the objects are nodes in a graph and the similarity is defined by the way neighbors interact, the clustering becomes the community detection [44, 84, 94] or the role extraction problem [93, 75]. A variety of community detection algorithms have been developed in recent years, such as the GN algorithm [80], the spectral modularity maximization algorithm [82], the Louvain method [12, 110], the Infomap algorithm [98], statistical inference [85], deep learning [120].

Graph-based clustering algorithms use the concepts and properties of graph theory, such that the clustering problem can be described as a graph partition problem. It is shown in [31] that, the graph partitioning problems including general weighted graph cuts, such as ratio association, ratio cut, normalized cut, and Kernighan-Lin objective function, can be formulated as an optimization problem

$$\min_{Y^T DY = I_q, Y^T Y is diagonal, Y \geq 0, \mathbf{1}_n \in \text{span}(Y)} -\text{trace}(Y^T DKDY), \tag{1.3}$$

where $K \in \mathbb{R}^{n \times n}$ is a symmetric kernel matrix and $D \in \mathbb{R}^{n \times n}$ is a diagonal matrix with all entries being positive. Letting X denote $D^{1/2}Y$, problem (1.3) can be reformulated into

$$\min_{X \in \mathcal{A}_v^{n,q}} -\text{trace}(X^T D^{1/2} K D^{1/2} X), \tag{1.4}$$

where v is a vector formed by the square roots of the diagonal entries in D, i.e., $v = \text{diag}(D^{1/2})$, and $\mathcal{A}_v^{n,q} = \{X \in \mathbb{R}^{n\times q} : X^TX = I_q, X \geq 0, v \in \text{span}(X)\}$, and $X \geq 0$ denotes that all entries of X are nonnegative.

1.2 Our Contribution

1.2.1 An Alternative Formulation of the Clustering Problem

By comparing the objective functions of problem (1.1), (1.2), and (1.4), it is easily seen that they have same pattern which is

$$\min_{X\in\mathcal{A}_v^{n,q}} f(X), \tag{1.5}$$

where $\mathcal{A}_v^{n,q} = \{X \in \mathbb{R}^{n\times q} : X^TX = I_q, X \geq 0, v \in \text{span}(X)\}$, and $X \geq 0$ denotes that all entries of X are nonnegative, $v \in \mathbb{R}^n$ is a vector with all entries being positive, and f is an application-dependent objective function. Note that, $\mathcal{A}_v^{n,q}$ is a general form of the set $\mathcal{A}_{n,q}$, and specifically, $\mathcal{A}_{n,q} = \mathcal{A}_{\mathbf{1}_n}^{n,q}$.

We formulate the community detection problem as a constrained nonsmooth optimization problem on the compact Stiefel manifold. It is proven that in an ideal graph, the global minimizer of $f : \mathcal{A}_{n,q} \to \mathbb{R} : X \mapsto -\text{trace}(X^TMX)$ is an assignment matrix that represents the ground truth, where

$$M = A - A\mathbf{1}_n\mathbf{1}_n^TA/(\mathbf{1}_n^TA\mathbf{1}_n), \tag{1.6}$$

which is called the modularity matrix [82], and where A is the adjacency matrix of the graph. In the presence of noise, the community detection is still formulated as the optimization problem

$$\min_{X\in\mathcal{A}_{n,q}} -\text{trace}(X^TMX), \tag{1.7}$$

under the assumption that the noise is not significant enough to change its minimizer.

We can see that (1.7) also has the same pattern as in (1.5). This formulation is one important contribution of this dissertation, and for more details see Chapter 3 and [116]. More contributions are generated based on this formulation.

Problem (1.5) can be reformulated by replacing the non-negative constraint $X \geq 0$ with a sparsity constraint $||X||_0 = n$ under condition that $f(X) = f(XD_i)$ for all $i = 1, .., q$, which yields

$$\min_{X\in\mathcal{B}_v} f(X), \tag{1.8}$$

where

$$\mathcal{B}_v = \{X \in \mathbb{R}^{n\times q} : X^T X = I_q, ||X||_0 = n, v \in \text{span}(X)\}, \tag{1.9}$$

$D_i = \text{diag}(1, ..., 1, -1, 1, ..., 1)$ whose i-th diagonal entry is -1, and $||X||_0$ denotes the total number of nonzero elements in X.

Problem (1.5) and Problem (1.8) are essentially equivalent in the sense that their solutions are connected, as is shown in the following lemma.

Lemma 1.2.1. *Consider Problem* (1.5) *and Problem* (1.8) *with the objective function f satisfying $f(X) = f(XD_i)$ for any $i = 1, .., q$, where $D_i = \text{diag}(1, ..., 1, -1, 1, ..., 1)$ whose i-th diagonal entry is -1. The following two statements hold:*

- *Let X be any matrix in $\mathcal{B}_v$. Then for any column of X, denoted by x_i, the signs of all nonzero entries in x_i are the same.*
- *Define a mapping $\psi : \mathbb{R}^{n\times q} \mapsto \mathbb{R}^{n\times q} : X \mapsto \hat{X} = XD_{j_1}D_{j_2}...D_{j_s}$, where $j_1, j_2, ..., j_s$ are the indices of the columns of X whose nonzero entries are all negative. Then X_* is a global minimizer of Problem* (1.8) *in the sense that $f(X_*) \leq f(Y), \forall Y \in \mathcal{B}_v$ if and only if $\psi(X_*)$ is a global minimizer of Problem* (1.5) *in the sense that $f(\psi(X_*)) \leq f(Z), \forall Z \in \mathcal{A}_v^{n,q}$.*

Proof. The first statement holds by the definition of $\mathcal{B}_v$. The second statement holds by the assumption of the function f. □

Due to the constraints of $\mathcal{B}_v$, the sparsest matrix in $\mathcal{B}_v$ has n nonzero entries. We reformulate Problem (1.8) and use one norm penalization to promote the sparsity of X, which yields a continuous optimization in (1.10). Problem (1.10) is exactly the form that we use for the proposed algorithms in this dissertation.Using one norm to promote sparsity on manifold has been widely used for the Stiefel manifold, see [57, 117]. If the minimizer of (1.10), denoted by X_*, is sufficiently close to $\mathcal{B}_v$, then one can find the closest matrix in $\mathcal{B}_v$ by a mapping $P_{\mathcal{B}_v}(X_*)$, see details in Lemma 1.2.2. If the i-th row j-th column of $P_{\mathcal{B}_v}(X_*)$ is not zero, then this implies that the i-th object is in the j-th cluster.

$$\min_{X\in\mathcal{F}_v} f(X) + \lambda||X||_1, \tag{1.10}$$

where $\lambda > 0$ is a tuning parameter, $\mathcal{F}_v = \{X \in \mathbb{R}^{n\times q} : X^T X = I_q, v \in \text{span}(X)\}$ is the domain, $v \in \mathbb{R}^n$ is a vector with all entries being positive, $\text{span}(X)$ denotes the column space of X and the gradient of f is Lipschitz continuous.

Lemma 1.2.2. *Let $v \in \mathbb{R}^n$ be a positive vector, W denote diag(v), Y denote a matrix in $\mathcal{B}_v$, d_i denote the number of nonzero entries in i-th column of Y, $u_i \in \mathbb{R}^{d_i}$ denote the vector forming by the nonzero entries of the i-th column of Y, and $u \in \mathbb{R}^n$ denote $(u_1^T\ u_2^T\ ...\ u_q^T)^T$. If $X_* \in \mathcal{F}_v$ is sufficiently close to Y, then it holds that*

$$Y = P_{\mathcal{B}_v}(X_*),$$

where $P_{\mathcal{B}_v}(X_*) = W P_{\mathcal{B}_{\mathbf{1}_n}}(W^{-1}X_*)$, $P_{\mathcal{B}_{\mathbf{1}_n}}(X_*) = (\frac{b_1}{||b_1 \odot v||}\ \frac{b_2}{||b_2 \odot v||}\ ...\ \frac{b_q}{||b_q \odot v||})$, $\odot$ denotes the Hadamard product, $b_j \in \mathbb{R}^n$ for $j = 1, 2, ..., q$ and

$$(b_j)_i = \begin{cases} \text{sign}((X_*)_{ij}) & if\ (X_*)_{ij}\ has\ the\ largest\ magnitude, \\ 0 & otherwise. \end{cases}$$

Proof. Without loss of generality, assume that Y has the form

$$Y = \text{diag}(u_1, u_2, ..., u_q) := \begin{pmatrix} u_1 & 0 & \cdots & 0 \\ 0 & u_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & u_q \end{pmatrix}$$

Partition the vector v by $v = (v_1^T\ v_2^T\ ...\ v_q^T)^T$, where $v_i \in \mathbb{R}^{d_i}$. It follows that $u_i = s_i v_i / ||v_i||, i = 1, 2, ...q$, where s_i is either 1 or -1. Therefore, $W^{-1}Y = \text{diag}(\frac{s_1}{||v_1||}\mathbf{1}_{d_1}, \frac{s_2}{||v_2||}\mathbf{1}_{d_2}, ..., \frac{s_q}{||v_q||}\mathbf{1}_{d_q})$. If $W^{-1}X_*$ is sufficiently close to $W^{-1}Y$ in the sense that the location of the largest magnitude entry of each row does not change, then it holds that

$$W^{-1}Y = P_{\mathcal{B}_{\mathbf{1}_n}}(W^{-1}X_*),$$

which implies $Y = P_{\mathcal{B}_v}(X_*)$. □

1.2.2 Relevant Riemannian Optimization Algorithms

Optimization over Riemannian manifolds has recently drawn much attention because of its application in many different fields. Almost all of the manifold optimization methods require computing the derivatives of the objective function and do not apply to the case where the objective function is nonsmooth. Chen et.al. proposed a Riemannian proximal gradient method called ManPG for a class of nonsmooth nonconvex optimization problems over a Stiefel manifold in [25], which is

$$\begin{aligned} &\min F(X) := f(X) + g(X), \\ &\text{s.t. } X \in \mathcal{M} := St(q, n) = \{X : X \in \mathbb{R}^{n \times q}, X^T X = I_q\}, \end{aligned} \tag{1.11}$$

where I_q denotes the $q \times q$ identity matrix ($q < n$), f is smooth, possibly nonconvex, and its gradient ∇f is Lipschitz continuous, g is convex, possibly nonsmooth, and is Lipschitz continuous and the proximal mapping of g is easy to find.

In [53], Huang and Wei extended the fast iterative shrinkage-thresholding (FISTA) algorithm to solve (1.11), and the resulting accelerated Riemannian manifold proximal gradient algorithm (AManPG) performed better than ManPG. The global convergence analysis of AManPG is presented in [53].

We can see that the cost function of the continuous optimization problem (1.10) has the same form of cost function (1.11) and the constraint of problem (1.10) is a subset of Stiefel manifold. To solve the continuous optimization problem (1.10), at first we propose the accelerated Riemannian manifold projected proximal gradient (ARPPG) method for the optimization problem (1.10). The idea of ARPPG is to solve a constrained nonsmooth optimization problem over a Stiefel manifold by deriving a projection from the Stiefel manifold $St(q, n)$ to the feasible set $\mathcal{F}_{\mathbf{1}_n} \subseteq St(q, n)$. The idea of ARPPG is natural if we do not know the manifold structure of the feasible set $\mathcal{F}_{\mathbf{1}_n}$. ARPPG works well for the community detection problem but there is no convergence analysis, see [116].

However, if a manifold structure of the feasible set $\mathcal{F}_{\mathbf{1}_n}$ can be proven, then we can use AManPG in [53] directly over the feasible set $\mathcal{F}_{\mathbf{1}_n}$ and the corresponding convergence theorem over $\mathcal{F}_{\mathbf{1}_n}$ can be derived. We show that the more general feasible set $\mathcal{F}_v$ is an embedded submanifold of $\mathbb{R}^{n \times q}$ and derive the optimization-related Riemannian geometry. The AManPG algorithm in [53] is used over the feasible set $\mathcal{F}_v$ for the clustering problems.

A key step in both the ARPPG and the AManPG method is to solve the proximal subproblem by using semi-smooth Newton method [119], [69]. The general idea of semi-smooth Newton method is to solve a system of nonlinear equations based on the generalized Jacobian. It is necessary to reduce the optimization problem to a system of nonlinear equations in order to use the semi-smooth Newton method. This can be obtained by considering the KKT conditions, see Chapter 3 and Chapter 4. The resulting algorithm however can be improved by solving the subproblem approximately instead of exactly. We propose such an approach in the I-AManPG method. A global convergence analysis of this novel method is given in Chapter 5.

There are many applications of the clustering problem, such as community detection, role extraction on networks, k-means model and discriminative k-means model and some graph parti-

tioning problems. The I-AManPG method is compared to the exact AManPG method and with existing algorithms on the community detection problem, see Chapter 6. Chapter 6 also includes empirical evaluation of I-AManPG on the k-means model, the discriminative k-means problem and the normalized cut problem.

The I-AManPG algorithm in this dissertation is different from the IRPG algorithm in [54] in the following aspects. The IRPG algorithm is motivated by a mathematical and theoretical question, which does not consider the performance. The IRPG algorithm assumes that the parameter μ in the proximal mapping is sufficiently small. It follows the step size one can be used and the global convergence analysis follows. A local convergence rate can be obtained based on a computationally expensive formulation of the Riemannian proximal mapping. The I-AManPG algorithm in this dissertation only requires μ to be positive. It follows that the search direction is descent and a line search algorithm can be used. The global convergence follows. The I-AManPG algorithm is more practical than the IRPG algorithm in [54].

While preparing this dissertation we were made aware of a conference paper [20], in which a manifold relaxation algorithm for k-means clustering was introduced using $\mathcal{F}_{\mathbf{1}_n}$ as the constraint set. The work in [20] and this dissertation were done independently. Carson et.al claimed that $\mathcal{F}_{\mathbf{1}_n}$ is a smooth manifold of $\mathbb{R}^{n\times q}$ in [20] but provided no proof while suggesting it would appear in the near future. We have found no such follow-up paper containing a proof. The authors also derived the tangent space, orthogonal projection onto the tangent space and a retraction. This dissertation contributes significantly more than this. The proof of the manfold structure of $\mathcal{F}_{\mathbf{1}_n}$ is not as simple as one might expect from [20] and the theory that is developed in Chapter 4 of this dissertation is more general. In particular, we rigorously prove that the Stiefel manifold with an all positive vector in its column space forms a compact embedded submanifold of $\mathbb{R}^{n\times q}$. Not only are proofs given, but the result is more general in the sense that the vector is not restricted to be the vector $\mathbf{1}_n$. A more efficient retraction is derived in this dissertation. The retraction in [20] requires to compute an exponential of a $n \times n$ matrix, which requires $O(n^3)$ operations and becomes is computationally unacceptable as n grows large. Our retraction is much more computationally approachable as it is only in the order of $O(nq^2)$. In addition, computationally efficient approaches to compute the intrinsic and extrinsic representation of tangent vectors are given in our dissertation. Such structure is not given in [20] as their algorithm is simple and does not need these

operations. Furthermore, the dissertation applies more sophisticated Riemannian algorithms to additional problems thus demonstrating and evaluating the applicability of the combination of the manifold $\mathcal{F}_{\mathbf{1}_n}$ and adaptations of a Riemannian proximal gradient can be efficient and competitive.

A potential limitation of the ARPPG, AManPG, and Inexact_AManPG algorithms is the requirement of the number of communities as an input. In practice, the number of communities q maybe unknown. The empirical evaluation considers when the number of communities parameter is taken smaller and larger than the actual number. To address this limitation, we also propose a recursive version of Inexact_AManPG algorithm. The algorithm and its empirical evaluation are presented in Chapter 7.

1.3 Overview and Dissertation Statement

This dissertation proposes the following thesis statement. The systematic exploration of the use of increasingly structured constraint geometry, the design of cost function alternatives to the standard problems exploiting the geometry and problem characteristics, algorithmic rigor and efficient computational design will produce an approach flexible enough to handle graph-based and Euclidean data-based optimization problems arising in a range of well-known clustering related problems. The approach will be a prime candidate for a new state-of-the-art and the starting point for further exploration of the efficient solution of related application problems.

The dissertation is organized as follows.

- In Chapter 2, we give a detailed description of the applications, namely, community detection, the k-means model, the discriminative k-means model, and the normalized cut problem. We show the connection between each application and Riemannian optimization problem and review existing algorithms for these applications.
- In Chapter 3, a new approach, the accelerated Riemannian projected proximal gradient method (ARPPG) is proposed to solve optimization problems with the appropriate constraints. The idea of ARPPG is to solve a constrained nonsmooth optimization problem over a Stiefel manifold by deriving a projection from the Stiefel manifold to the feasible set $\mathcal{F}_{\mathbf{1}_n}$ which is a subset of Stiefel manifold. ARPPG is applied to the community detection problem, and the numerical results show that ARPPG is comparable with existing algorithms for community detection.
- We show that the feasible set $\mathcal{F}_v$ is a Riemannian submanifold in Chapter 4 , and derive the optimization-related Riemannian geometry. An accelerated Riemannian manifold proximal

gradient algorithm (AManPG algorithm) [53] is adapted using the geometry of $\mathcal{F}_v$'s objects for the clustering problems.

- The I-AManPG method is proposed in order to improve the efficiency of AManPG and its global convergence analysis is presented in Chapter 5.
- The I-AManPG algorithm is empirically evaluated in Chapter 6. Four applications, community detection, the k-means model, the discriminative k-means model, and the normalized cut problem are considered.
- In Chapter 7, we propose a recursive version for I-AManPG algorithm to address the potential limitation on the number of communities parameter and to improve the efficiency when the number of clusters q is large.
- A summary of completed and future work is given in Chapter 8.
- Finally, basic concepts of Riemannian optimization, an overview of projected proximal gradient method on Euclidean space and basic elements of graph theory are reviewed and tutorial references given in the appendices A, B, and C.

CHAPTER 2

THE APPLICATIONS AND THE RIEMANNIAN FORMULATIONS

In this chapter, we describe the applications, community detection, k-means model, discriminative k-means model and normalized cut graph partitioning problem. For each application a general description is given followed by a review of relevant algorithms. Finally, we show the connection between each application and the clustering problem (1.5).

2.1 Community Detection

2.1.1 Description of Community Detection Problem

Networks are a natural representation of various kinds of complex systems, where networks are sets of nodes or vertices joined together in pairs by links or edges. There are a number of networks. One common network is Facebook, which is a large social network, where more than one billion people are connected via virtual acquaintanceship. Another well-known example is the Internet, the physical network of computers, routers and modems which are linked via cables or wireless signals. Many other examples come from biology, physics, engineering, computer science, ecology, economics, marketing, etc.

A number of recent studies have focused on the statistical properties of networked systems. One common property of many networks is called the community structure [84], which is the division of network nodes into groups within which the network connections are dense, but between which are sparser. These groups are called communities, or modules. An example of a network with such a community structure is shown in Figure 2.1.

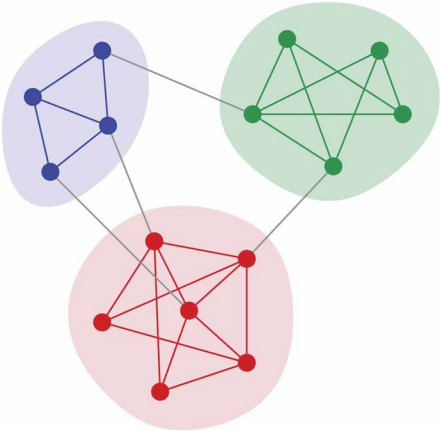


Figure 2.1: A small network with 3 communities, which have dense internal links but between which there is only a lower density of external links.

Detecting community structure in a network can provide powerful help in understanding and exploiting the structure of networks, and it has various practical applications [44]. Communities in a social network might represent real social groupings, perhaps by acquaintanceship, interest or background; communities in a citation network might represent related papers on a single topic; communities in a metabolic network might represent cycles and other functional groupings; communities on the web might represent pages on related topics.

Numerous community detection algorithms have been developed suing a wide variety of techniques, e.g., removal of high-betweenness edges [44], modularity optimization [84] [94], random walk [125], statistical inference [85].

2.1.2 Some Existing Algorithms for Community Detection

GN Algorithm. The GN algorithm is the one of most popular divisive algorithms and was proposed by Girvan and Newman [84, 44]. The philosophy of divisive algorithms is to identify communities in a graph by repeatedly removing edges with lowest similarity metric, thus moving from all nodes in one community to repeatedly partitioning current communities into multiple smaller ones until an acceptable community assignment is found.

The GN algorithm is historically important, because it marked the beginning of a new era in the field of community detection. The GN algorithm assumes the graphs are undirected and unweighted, but generalizations to more complicated network types are possible [79]. The GN algorithm follows

roughly the philosophy of divisive algorithms, but it differs slightly from them. The GN algorithm does not remove the edges with the lowest similarity, but it rather finds the highest "betweenness", where betweenness is a measure that favors edges that lie between communities and disfavors those that lie inside communities.

The simplest and most efficient edge betweenness measure is the shortest-path betweenness. Freeman [43] first proposed the betweenness centrality of a vertex i which is defined as the number of shortest paths between pairs of other vertices that run through i. The definition of the betweenness centrality of a vertex assumes that the graphs are undirected. Inspired by this, Girvan and Newman generalized the vertex betweenness to edge betweenness as the number of shortest paths between pairs of other vertices that run through the edge. If a network contains communitities that are only loosely connected by a few intercommunity edges, then all shortest paths between different communities must go along one of these few edges. Thus, the edges connecting communities will have high edge betweenness. By removing these edges, communities can be separated from one another and so reveal the underlying community structure of the graph.

In the following Figure 2.2, the thick edge in the middle has a much higher betweenness than all other edges [42], because all shortest paths connecting vertices of the two communities run through it.

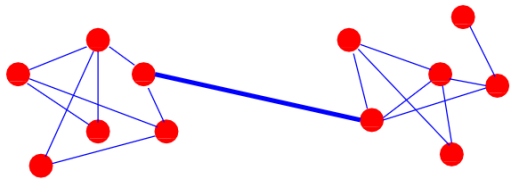


Figure 2.2: The thick edge has the highest edge betweenness, see [42].

The steps of the divisive first phase of GN algorithm are:

Step 1: Calculate betweenness scores for all edges in the network.

Step 2: Find and remove the edge with the highest betweenness score.

Step 3: Recalculate betweenness for all remaining edges.

Step 4: Repeat from Step 2 until a network is decomposed into desired number of communities.

The algorithm runs in worst-case time $O(m^2n)$, (or $O(n^3)$ on a sparse graph) for a graph with m edges and n vertices: calculating the betweenness of all edges of the graph can be obtained in a time that scales as $O(mn)$ [84], (or $O(n^2)$ on a sparse graph) using a technique based on breadth-first search [17].

The set of divisions found in the divisive first phase of GN algorithm is represented by a dendrogram. How do we know when the communities found by the algorithm are good ones? In order to get the good divisions of the network, communities, the dendrogram must be cut at some level. Newman and Girvan [84, 27] define modularity as a measure of the quality of a particular division of a network, given by

$$Q = \frac{1}{2m} \sum_{i,j} (A_{ij} - \frac{k_i k_j}{2m}) \delta(\sigma_i, \sigma_j) \tag{2.1}$$

where A_{ij} is an element of the adjacency matrix of an undirected unweighted graphs, and A_{ij} is 1 or 0 which depends on if vertices i and j are connected, and $k_i = \sum_j A_{ij}$ which is called the degree of vertex i. The definition of the modularity can be generalized trivially to weighted graphs in which each edge has a numeric strength associated with it, by making the values of the matrix A_{ij} equal to those weights, rather that just 1 or 0.

The range of modularity is between $-\frac{1}{2}$ and 1 [111] [18], but in order to have a modularity structure for a network the modualrity must be positive. The higher Q, the better the division. Typically, as an additional refinement to the GN algorithm, one can calculate Q for each division of a network into communities by moving down the dendrogram, and looking for local peaks in the value of Q, which indicate particularly satisfactory divisions.

Danon et al.'s Algorithm. The first algorithm derived to maximize modularity was a greedy method of Newman [80]. It is an agglomerative hierarchical clustering method, where groups of vertices are successively joined to form larger communities such that modularity increases after the merging. One starts from n clusters, each containing a single vertex. Edges are not initially present, they are added one by one during the procedure. However, the modularity of partitions explored during the procedure is always calculated from the full topology of the graph, as we want to find the modularity maximum on the space of partitions of the full graph. Adding a first edge to the set of disconnected vertices reduces the number of groups from n to $n-1$, so it delivers a new partition of the graph. The edge is chosen such that this partition gives the maximum increase

(minimum decrease) of modularity with respect to the previous configuration. All other edges are added based on the same principle. The number of partitions found during the procedure is n, each with a different number of clusters, from n to 1. The largest value of modularity in this subset of partitions is the approximation of the modularity maximum given by the algorithm. At each iteration step, one needs to compute the variation ΔQ of modularity given by the merger of any two communities of the running partition, so that one can choose the best merger. However, merging communities between which there are no edges can never lead to an increase of Q, so one has to check only the pairs of communities which are connected by edges, of which there cannot be more than m. Since the calculation of each ΔQ can be done in constant time, this part of the calculation requires a time $O(m)$. After deciding which communities are to be merged, one needs to update the matrix e_{ij} expressing the fraction of edges between clusters i and j of the running partition (necessary to compute Q), which can be done in a worst-case time $O(n)$. Since the algorithm requires $n-1$ iterations (community mergers) to run to completion, its complexity is $O((m+n)n)$(, or $O(n^2)$ on a sparse graph).

Newman's fast greedy algorithm [80] heavily depends on the community size distribution, showing a tendency to find large communities at the expense of smaller ones. In order to treat the clusters of different sizes equally, Danon et al. [28] suggested normalizing the modularity variation ΔQ produced by the merger of two communities by the fraction of edges incident to one of the two communities [28]. Compared to Newman's fast greedy algorithm, this technique leads to better modularity optima, especially when communities are very different in size and is comparable in speed. In this dissertation, Danon's algorithm is used as the representative greedy algorithm instead of Newman's fast greedy algorithm, because Danon's algorithm often finds a better modularity, while maintaining the same complexity as Newman's algorithm. Both Newman's fast greedy algorithm and Danon's algorithm are able to be applied to weighted networks.

Infomap. The Infomap method was introduced by Rosvall and Bergstrom [98] and has been shown to be a quite successful method for community detection [64]. It is based on the information and optimal coding theory and it can be applied to weighted and directed networks. The infomap method uses the probability flow of random walks on a network as a proxy for information flows in the real system and decompose the network into modules by compressing a description of the probability flow.

Rosvall and Bergstrom introduced the map equation based on the fact that a random walker should spend most of its time within communities and rarely use edges across communities [98]. One way to describe the trajectory of a random walker is to assign a specific codeword to each node of the graph. In this case, one can compute the expected length of a codeword as the entropy of the stationary frequency distribution to visit each node. The expected codeword length of a single step of a random walk is bounded by the Shannon entropy [102].

Finding a code that reaches this lower bound is very complicated. To reduce this complexity, the map equation uses a two-level code [98]. The first level encodes the different communities while the second level encodes the individual nodes. This allows the reuse of the same codewords for nodes in different communities. This reduces the expected length of each step of the random walk. So detecting an accurate community division becomes finding an optimal code to minimize the expected length of each step of a random walker, which is given by the map equation

$$L(\sigma) = (1-\rho)H_q + \sum_c q_c H_c, \tag{2.2}$$

where $1-\rho := \sum_c (1-\rho_c) = \sum_c (1 - \frac{\sum_{ij} A_{ij}\delta(\sigma_i,\sigma_j)}{\sum_i k_i \delta(\sigma_i,c)})$ is the probability of switching communities, $q_c = \sum_i \pi_i \delta(\sigma_i, c)$ is the probability a certain community c is visited, where π_i are the stationary probabilities of the random walker, $H_q = -\sum_c q_c \log q_c$ is the average code length that the random walker exists from a community, and $H_c = -\sum_i \frac{\pi_i}{q_c+(1-\rho_c)} \log \frac{\pi_i}{q_c+(1-\rho_c)} - \frac{1-\rho_c}{q_c+(1-\rho_c)} \log \frac{1-\rho_c}{q_c+(1-\rho_c)}$ is the entropy for moving within a community c.

To implement the infomap method [98], one can repeatedly merge pairs of communities which produce the largest decrease in the map equation, and then use simulated annealing [62] to refine the partition.

Louvain Algorithm. According to the results of Lancichinetti and Fortunato [64], the Louvain method [12] is known to be one of the most effective and efficient algorithms for the modularity optimization.

The Louvain algorithm [12] is a heuristic method that is based on the modularity optimization. It has two phases which are repeated iteratively. Assume that one starts with a weighted network of N nodes. The specific algorithm is as follows, and Figure 2.3 shows the visualization of the steps.

- Phase 1: Modularity Greedy Optimization

– Step 1: Initialize partition—we assign a different community to each node of the network, so there are N communities initially;

– Step 2: For each node i, we consider the neighbours j of i and evaluate the gain of modularity by removing i from its community and by placing it in the community of j.

– Step 3: We place node i in the community which has a local maxima of the gain of modularity. If the gain is not positive, i stays in its original community.

This first phase stops when no individual moves can improve the modularity. The complexity of each local move in Phase 1 is in $O(\#of\ neighbors)$.

- Phase 2: Community Aggregation

  – Step 1: We get the new network whose nodes are the communities found during the first phase.

  – Step 2: Get each element of the new aggregated weighted adjacency matrix $\tilde{A}_{cd}$ between the new nodes c and d by adding the number of the links between nodes in the corresponding two communities c and d in the original graph.

- Reapply the first phase of the algorithm to the resulting weighted network and iterate until no more changes and a maximum of modularity (or a minimum of cost function $\mathscr{H}$) is attained.

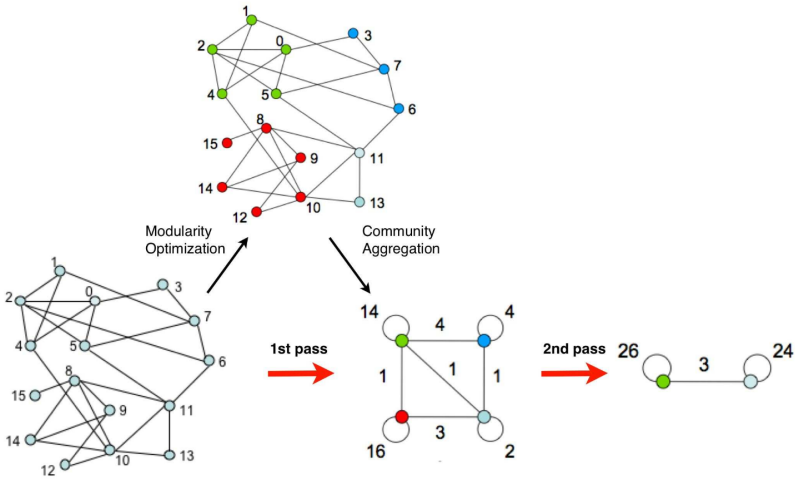


Figure 2.3: Visualization of the steps of Louvain Method, see [12].

The pseudocode for Louvain method is given in Algorithm 1 [106], where function Greedy(G) can be implemented in the following Algorithm 2.

Algorithm 1 Louvain Algorithm

function Louvain(Graph G)
    $\sigma \leftarrow$Greedy(G)
    $\Sigma \leftarrow \sigma$
while increase in modularity **do**
    $G \leftarrow$Aggregate(G, Σ)
    $\Sigma \leftarrow$Greedy(G)
    $\sigma_i \leftarrow \Sigma_{\sigma_i}$ for all nodes i
end whilereturn σ
end function

Algorithm 2 Phase 1—Greedy Algorithm

function Greedy(Graph G)
    Initialize $\sigma \leftarrow i$ for each node i
while increase in modularity **do**
    for each node i **do**
        $C \leftarrow \{\sigma_j | (i,j) \in E\} \bigcup \sigma_i$
        for each community $d \in C$ **do**
            $\Delta_d \leftarrow \Delta(-\mathscr{H}(\sigma_i = c \longmapsto d))$
        end for
        $\sigma_i \leftarrow argmax_d \Delta_d$
    end for
end whilereturn σ
end function

Traag et al. in [110] show that the Louvain algorithm has major defect that the Louvain algorithm may yield arbitrarily badly connected communities.To address this problem, they introduce the Leiden algorithm and prove that the Leiden algorithm yields communities that are guaranteed to be connected.

Newman's spectral optimization method. Graph partitioning is often done using the spectral method based on the Laplacian matrix. The Laplacian matrix is defined as the difference between the degree matrix and the adjacency matrix. The spectral method uses the first k eigenvectors corresponding to the k smallest eigenvalues of the Laplacian matrix and clusters the graph nodes by using for example k-means clustering based on the first k eigenvectors. In [82], Newman proposed the spectral method for community detection by replacing the Laplacian matrix with the modularity matrix. Modularity can be optimized using the eigenvalues and eigenvectors of the modularity matrix M as defined in (1.6). The Newman's spectral method computes the eigenvector, u_1, of the modularity matrix corresponding to the largest positive eigenvalue, λ_1, and groups the vertices according to the signs of the component of u_1 to divide the network into two clusters. This process is repeated for each of the parts using the generalized modularity matrix $M^{(g)}$, where $M_{ij}^{(g)} = M_{ij} - \delta_{ij} \sum_{k\in \text{ group } g} B_{ik}$. For the single split, the complexity is $O((m+n)n)$, and the running time depends on the dendrogram produced by the repeated splittings. The average depth of the dendrogram is $\log n$, giving an average running time for the whole algorithm of $O((m+n)n \log n)$. It is important to note that modularity must be always computed from the full adjacency matrix of the original graph. The drawback of the method is similar to that for spectral bisection, i.e., the algorithm gives the best results for bisections, whereas it is less accurate when the number of communities is larger than two [40].

2.1.3 The Connection Between the Community Detection and Optimization Problem in Terms of the Assignment Matrix

In [116], it is proven that in an ideal graph, the global minimizer of $f : \mathcal{A}_{n,q} \to \mathbb{R} : X \mapsto -\text{trace}(X^T M X)$ is an assignment matrix that represents the ground truth, where $M = A - A\mathbf{1}_n\mathbf{1}_n^T A/(\mathbf{1}_n^T A \mathbf{1}_n)$ is the modularity matrix and A is the adjacency matrix of the graph. In the presence of noise, the community detection is stilled formulated as the optimization problem

$$\min_{X\in\mathcal{A}_{n,q}} -\text{trace}(X^T M X), \tag{2.3}$$

under the assumption that the noise is not significant enough to change its minimizer.

As discussed in Section 1.2.1, we reformulate Problem (1.8) and use one norm penalization to promote the sparsity of X, which yields a continuous optimization in (1.10). So when we use the algorithms, ARPPG, I-AManPG, and the recursive I-AManPG, the cost function for community detection is

$$\min_{X \in \mathcal{F}_{\mathbf{1}_n}} -\text{trace}(X^T M X) + \lambda ||X||, \tag{2.4}$$

where $\lambda > 0$ is the tuning parameter, $\mathcal{F}_{\mathbf{1}_n} = \{X \in \mathbb{R}^{n \times q} : X^T X = I_q, \mathbf{1}_n \in \text{span}(X)\}$ is the domain.

2.2 k-means Model

2.2.1 Description of k-means Model

k-means clustering [70, 39, 47] is a method of vector quantification, originally from signal processing, that aims to partition n observations into $k(\leq n)$ clusters in which each observation belongs to the cluster with the nearest mean serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells [5].

Formally, the objective of k-means clustering is to partition the n observations into $k(\leq n)$ sets $\{S_1, S_2, ..., S_k\}$ so as to minimize the within-cluster sum of squares:

$$\underset{S}{\text{argmin}} \sum_{i=1}^{k} \sum_{x \in S_i} ||x - \mu_i||, \tag{2.5}$$

where μ_i is the mean of points in S_i. This is equivalent to minimizing the pairwise squared deviations of points in the same cluster

$$\underset{S}{\text{argmin}} \sum_{i=1}^{k} \frac{1}{2|S_i|} \sum_{x,y \in S_i} ||x - y||^2. \tag{2.6}$$

2.2.2 The Standard Algorithm for the k-means Model

The most common algorithm uses an iterative refinement technique. It is often called "the k-means algorithm". Given an initial set of k means $m_1^{(1)}, ..., m_k^{(1)}$, the algorithm proceeds by alternating between two steps [72]:

- **Assignment Step at t:** Assign each observation to the cluster with the nearest mean: that with the least squared Euclidean distance

$$S_i^{(t)} = \{x_p : ||x_p - m_i^{(t)}||^2 \leq ||x_p - m_j^{(t)}||^2, \ \forall j, 1 \leq j \leq k\}, \tag{2.7}$$

where each x_p is assigned to exactly one $S^{(t)}$, even if it could be assigned to two or more of them. To break ties consistently, e.g., one can assign a observation to the cluster with the lowest index if there are several equidistant centroids.

- **Update Step:** Recalculate means for observations assigned to each cluster

$$m_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j \tag{2.8}$$

The algorithm is often presented as assigning objects to the nearest cluster by distance. Using a different distance function other than (squared) Euclidean distance may prevent the algorithm from converging. Various modifications of k-means such as spherical k-means [32] and k-medoids [76] have been proposed to allow using other distance measures.

2.2.3 Initialization Method for k-means Model

The algorithm does not guarantee convergence to the global optimum. The result may depend on the initial means. Commonly used initialization methods are Forgy and Random Partition [46]. The Forgy method randomly chooses k observations from the dataset and uses these as the initial means. The Random Partition method first randomly assigns a cluster to each observation and then proceeds to the update step, thus computing the initial mean to be the centroid of the cluster's randomly assigned points. The Forgy method tends to spread the initial means out, while Random Partition places all of them close to the center of the data set. According to Hamerly et al., [46] the Random Partition method is generally preferable for algorithms such as the k-harmonic means and fuzzy k-means. For expectation maximization and standard k-means algorithms, the Forgy method of initialization is preferable. A comprehensive study by Celebi et al.,[21] however, found that popular initialization methods such as Forgy, Random Partition, and Maximin often perform poorly, whereas Bradley and Fayyad's approach [16] performs consistently well and k-means++ [112] performs generally well. For the k-means clustering algorithm in this dissertation, the k-means++ algorithm [112] is used to generate the initial means.

2.2.4 The Connection Between k-means Model and Optimization Problem in Terms of the Assignment Matrix

The k-means model can be rewritten as an alternating minimization algorithm for solving the optimization problem [15]

$$\min_{X \in \mathcal{A}_{n,q}} ||A - XX^T A||_F^2, \tag{2.9}$$

where A is a data matrix where each data point in $\mathbb{R}^d$ corresponds to a row of A. To see this, we notice that $||A - XX^T A||_F^2 = \sum_{i=1}^n ||A_{(i)} - X_{(i)} X^T A||_2^2$, where $A_{(i)}$ and $X_{(i)}$ denote the i-th rows of A and X, respectively, and $X_{(i)} X^T A$ denotes the centroid of the cluster the i-th data point belongs to [15].

If we fix the assignment matrix X, we take the gradient of $||A - XX^T A||_F^2 = ||A - XC||_F^2$ with respect to C, which is $-2X^T(A - XC) = 0$. So, we can get $X^T XC = X^T A$. $C = (X^T X)^{-1} X^T A$ is the centriod of the cluster. This is the update step in the classical k-means clustering. If we fix the centroid, then we need to solve the optimization problem with respect to X, which corresponds to the assignment step in the classical k-means clustering. In C, there is a term $(X^T X)^{-1}$, if we consider the $X \in St(q, n)$, then we can get the cost function in (2.9).

Note that, when we use the algorithms, ARPPG, I-AManPG, and the recursive I-AManPG, the cost function for the k-means model can be reformulated as

$$\min_{X \in \mathcal{F}_{\mathbf{1}_n}} ||A - XX^T A||_F^2 + \lambda ||X||, \tag{2.10}$$

where $\lambda > 0$ is the tuning parameter, $\mathcal{F}_{\mathbf{1}_n} = \{X \in \mathbb{R}^{n \times q} : X^T X = I_q, \mathbf{1}_n \in \text{span}(X)\}$ is the domain.

2.3 Discriminative k-means Model

2.3.1 Description of Discriminative k-means Model

When the data sets live in a very high dimensional space or the number of attributes are larger than they need to be, the dimension reduction can be used. In [33], the dimensionality reduction technique, Linear Discriminant Analysis (LDA), is combined with the k-means clustering. It is further analyzed in [121] and is shown to be equivalent to a kernel k-means. The resulting optimization problem is given therein by

$$\min_{X \in \mathcal{A}_{n \times q}, \lambda_{regu} > 0} \text{trace}(X^T (I_n + \frac{1}{\lambda_{regu}} A^T A)^{-1} X) + \log\det(I_n + \frac{1}{\lambda_{regu}} A^T A), \tag{2.11}$$

where $\lambda_{regu} > 0$ is the regularization parameter. This problem is solved by alternating between the computation of X for a given λ_{regu} and the computation of λ_{regu} for a given X. This first subproblem of the computation of X for a given λ_{regu} is in the form of (1.5).

2.3.2 Existing Algorithms for Discriminative k-means Model

The Discriminative K-menas (DisKmeans) algorithm is discussed by Ye et al. in [121]. They show that the LDA projection can be factored out from the integrated LDA subspace selection and clustering formulation. This results in a simple trace maximization problem associated with a regularized Gram matrix of the data. The solution to this trace maximization problem leads to the DisKmeans algorithm for simultaneous LDA subspace selection and clustering. DisKmeans is shown to be equivalent to kernel k-means, where discriminative subspace selection essentially constructs a kernel Gram matrix for clustering.

The locally linear embedding (LLE) in [99] and the Laplacian eigenmaps (LEI) in [10] are another two representative algorithms for discriminative clustering. The LLE algorithm is based on geometric intuition. Firstly, it assigns neighbors to each data point (for example by using the k nearest neighbors). Then it computes the weights that best linearly reconstruct the data point from its neighbors solving the constrained least-square problem. The constrained weights that minimize the reconstruction errors obey the property of symmetry. Based on this, LLE constructs a neighborhood-preserving mapping that maps the high-dimensional coordinates of each neighborhood to global internal coordinates on the manifold of lower dimension. Finally, it computes the low-dimensional embedding coordinates by solving a sparse eigenvalue problem.

The LEI algorithm is also a dimensionality reduction algorithm. The LEI algorithm uses the properties of the Laplace Beltrami operator to construct invariant embedding maps. LEI constructs the adjacency graphs and chooses the weights for the edges by considering the heat kernel. Then, it computes eigenvalues and eigenvectors of Laplacian matrix and uses the eigenvectors of the d smallest eigenvalues for embedding in a d-dimensional Euclidean space.

2.3.3 The Connection Between the Discriminative k-means Model and Optimization Problem in Terms of Assignment Matrix

When we use the algorithms, ARPPG, I-AManPG, and the recursive I-AManPG, the cost function for the discriminative k-means model (2.11) can be reformulated as

$$\min_{X\in\mathcal{F}_{\mathbf{1}_n},\lambda_{regu}>0} \text{trace}(X^T(I_n+\frac{1}{\lambda_{regu}}A^TA)^{-1}X)+\log\det(I_n+\frac{1}{\lambda_{regu}}A^TA)+\lambda||X||, \tag{2.12}$$

where $\lambda>0$ is the tuning parameter, $\mathcal{F}_{\mathbf{1}_n}=\{X\in\mathbb{R}^{n\times q}: X^TX=I_q, \mathbf{1}_n\in\text{span}(X)\}$ is the domain.

2.4 Normalized Cut

Normalized cut has been widely used for image segmentation which can be treated as a graph partitioning problem. The normalized cut criterion is a global criterion for partitioning a graph. Normalized cut measures both the total dissimilarity between the different groups as well as the total similarity within the groups [103]. The normalized cut problem is to optimize the normalized cut criterion.

It is shown in [31] that some graph partitioning problems including general weighted graph cuts, such as ratio association, ratio cut, Kernighan-Lin objective, and normalized cut, can be formulated as an optimization problem as shown in (1.3). These graph partitioning problems have been useful in many areas, such as circuit layout [22] and image segmentation [103].

Let X denote $D^{1/2}Y$. It follows that $X\in\mathcal{A}_v^{n,q}$, where v is a vector formed by the square roots of the diagonal entries in D, i.e., $v=\text{diag}(D^{1/2})$. Therefore, Problem (1.3) can be reformulated into

$$\min_{X\in\mathcal{A}_v^{n,q}} -\text{trace}(X^TD^{1/2}KD^{1/2}X), \tag{2.13}$$

where $K\in\mathbb{R}^{n\times n}$ is a symmetric kernel matrix. Polynomial kernel, Gaussian kernel and Sigmoid kernel are some popular kernel functions [31]. In this dissertation, we use Gaussian kernel for the normalized cut problem.Note that Problem (2.13) is in the form of (1.5).

2.4.1 Description of Normalized Cut

Taking $W=DKD$ in Problem (2.13), the optimization formulation for normalized cut problem is given by

$$\min_{X\in\mathcal{A}_v^{n,q}} f_{\text{NC}}(X)=-\text{trace}(X^TD^{-1/2}WD^{-1/2}X), \tag{2.14}$$

where $\mathcal{A}_v^{n,q} = \{X \in \mathbb{R}^{n\times q} : X^T X = I_q, X \geq 0, v \in \text{span}(X)\}$, and $X \geq 0$ denotes that all entries of X are nonnegative.

To read the images, we convert the color images to gray images which are read using the function "imread" built in Matlab, and stored as a two-dimensional array. In the case of gray image segmentation, the matrix $W \in \mathbb{R}^{mn\times mn}$ is an affinity matrix of an m by n image of grey pixels, $D \in \mathbb{R}^{mn\times mn}$ is a diagonal matrix with $D_{ii} = \sum_{j=1}^{mn} W_{ij}$, and $v = \text{diag}(D^{1/2})$. In this dissertation, we use the approach in [105] to choose W and D, to further shift the diagonal entries of W and D by a constant.

Problem (2.14) can be optimized by the weighted kernel k-means algorithm, see e.g., [31, Algorithm 1]. Note that Problem (2.14) has many low-quality local minimizers and descent algorithms are usually not able to escape from them. Thus, initialization plays an important role in finding an acceptable solution. Let U be the $n \times q$ matrix of the q leading eigenvectors of the matrix $D^{-1/2}WD^{-1/2}$. If X is only required to be orthonormal, then U is a global minimizer of (2.14). Since U is unlikely to be in $\mathcal{A}_v^{n,q}$, one approach is to find a matrix in $\mathcal{A}_v^{n,q}$ that is close to U. Different notions of closeness yield different methods.

2.4.2 Initialization Methods for Normalized Cut

Bach and Jordan [6] seek to find a matrix $Y \in \mathcal{A}_v^{n,q}$ that minimizes

$$\|UU^T - YY^T\|_F. \tag{2.15}$$

In other words, the difference between U and Y is measured by the difference between the two orthogonal projection matrices. The weighted kernel k-means is suggested to solve (2.15) see [6, Figure 1].

Shi and Malik [103] propose to find an indicator matrix that is closest to U up to a rotation, where an indicator matrix Z is defined by $Z \geq 0, Z^T Z$ is diagonal, and each row of Z has an entry being one. Specifically, let $\tilde{U}$ denote the matrix formed by normalizing all rows of U. The task is to find an indicator matrix Z and a q-by-q orthonormal matrix Q that minimize

$$\|Z - \tilde{U}Q\|_F.$$

Karypis and Kumar [60] give a fast, multi-level graph partitioning algorithm that produces equally-sized clusters and is called METIS. It is shown to be an effective method for the kernel k-means initialization.

2.4.3 The Connection Between the Normalized Cut and Optimization Problem in Terms of the Assignment Matrix

We propose to initialize the weighted kernel k-means algorithm by solving a variant of problem (1.5).

$$\min_{X \in \mathcal{A}_v^{n,q}} f_{\mathrm{NC}}(X) = -\mathrm{trace}(X^T D^{-1/2} W D^{-1/2} X), \tag{2.16}$$

where $\mathcal{A}_v^{n,q} = \{X \in \mathbb{R}^{n \times q} : X^T X = I_q, X \geq 0, v \in \mathrm{span}(X)\}$, and $X \geq 0$ denotes that all entries of X are nonnegative. Here, $\mathcal{A}_v^{n,q}$ is a general form of the set $\mathcal{A}_{n,q}$ in (1.5), $\mathcal{A}_{n,q} = \mathcal{A}_{\mathbf{1}_n}^{n,q}$. Such clusters are then used as initializations for the weighted kernel k-means algorithm.

Note that, when we use the algorithms, ARPPG, I-AManPG, and the recursive I-AManPG, the cost function for the normalized cut problem can be reformulated as

$$\min_{X \in \mathcal{F}_v} -\mathrm{trace}(X^T D^{-1/2} W D^{-1/2} X) + \lambda ||X||, \tag{2.17}$$

where $\lambda > 0$ is the tuning parameter, $\mathcal{F}_v = \{X \in \mathbb{R}^{n \times q} : X^T X = I_q, v \in \mathrm{span}(X)\}$ is the domain.

CHAPTER 3

ACCELERATED RIEMANNIAN PROJECTED PROXIMAL GRADIENT OPTIMIZATION METHOD ON COMMUNITY DETECTION

3.1 Accelerated Riemannian Projected Proximal Gradient Method

In [53], Huang and Wei generalized FISTA [9] from the Euclidean space to the Riemannian setting and considered the general nonconvex optimization problem

$$\min_{X\in\mathcal{M}} F(X) = f(X) + g(X), \tag{3.1}$$

where $\mathcal{M} \subset \mathbb{R}^{n\times q}$ is a Riemannian submanifold, $f : \mathbb{R}^{n\times q} \to \mathbb{R}$ is L-continuously differentiable (may be nonconvex) and g is continuous and convex but may not be differentiable.

Inspired by [53], we can solve the constrained nonconvex optimization problem by introducing a projection. The general form can be presented as

$$\min_{X\in C\subseteq\mathcal{M}} F(X) = f(X) + g(X), \tag{3.2}$$

where $\mathcal{M} \subset \mathbb{R}^{n\times q}$ is a Riemannian submanifold, $f : \mathbb{R}^{n\times q} \to \mathbb{R}$ is L-continuously differentiable (may be nonconvex) and g is continuous and convex but may not be differentiable.

For the clustering problems, if $\mathcal{M}$ is the Stiefel manifold $St(q, n)$ adding the constraint $\mathbf{1}_n \in \mathcal{R}(X)$ defines a feasible set $\mathcal{F}_{\mathbf{1}_n} = \{X \in St(q, n),\ \ \mathbf{1}_n \in \mathcal{R}(X)\} \subseteq St(q, n)$.

The optimization problem (3.2) can be solved by modifying the AManPG method to the accelerated Riemannian manifold projected proximal gradient method (ARPPG) by adding the projection (3.4) derived in the next section. ARPPG is in Algorithm 3.

There are several retractions that can be constructed for the Stiefel manifold. Algorithm 3, uses the efficient retraction in [53] based on the singular value decomposition (SVD):

$$[Q, R] = \mathrm{qr}(X + \eta_X),\ [U, S, V] = \mathrm{svd}(R),$$
$$R_X(\eta_X) = Q(UV^T),$$

Algorithm 3 Accelerated Riemannian Manifold Projected Proximal Gradient Method(ARPPG)

Input: Lipschitz constant L on ∇f, parameter $\mu \in (0, 1/L]$ in the proximal mapping, line search parameter $\sigma \in (0, 1)$, shrinking parameter in line search $\beta \in (0, 1)$, positive integer N for safeguard;

$t_0 = 1, y_0 = x_0, z_0 = x_0; \lambda = \lambda_0$
for $k = 0, ...$ **do**
  if mod(k, N) = 0 **then** ▷ Invoke safeguard every N iterations
    Invoke Algorithm 4: $[z_{k+N}, x_k, y_k, t_k] = Algo4(z_k, x_k, y_k, t_k, F(x_k))$;
  end if
  Compute

$$\eta_{y_k} = \underset{\eta \in T_{y_k}\mathcal{M}}{\operatorname{argmin}} \langle \operatorname{grad} f(y_k), \eta \rangle + \frac{1}{2\mu}||\eta||_F^2 + g(y_k + \eta);$$

  $x_{k+1} = R_{y_k}(\eta_{y_k})$;
  $x_{k+1} = proj(x_{k+1})$;
  $t_{k+1} = \frac{\sqrt{4t_k^2+1}+1}{2}$;
  Compute

$$y_{k+1} = R_{x_{k+1}}(\frac{1 - t_k}{t_{k+1}} R_{x_{k+1}}^{-1}(x_k));$$

  Compute $y_{k+1} = proj(y_{k+1})$.
end for
$X_* = x_{k+1}$

Algorithm 4 Safeguard for Algorithm ARPPG

Input: $[z_k, x_k, y_k, t_k, F(x_k)]$;

Output: $[z_{k+N}, x_k, y_k, t_k]$;

Compute
$$\eta_{z_k} = \underset{\eta \in T_{z_k}\mathcal{M}}{\operatorname{argmin}} \langle \operatorname{grad} f(z_k), \eta \rangle + \frac{1}{2\mu}||\eta||_F^2 + g(z_k + \eta);$$
Set $\alpha = 1$;
while $F(proj(R_{z_k}(\alpha\eta_{z_k}))) > F(z_k) - \sigma\alpha||\eta_{z_k}||_F^2$ **do**
    $\alpha = \beta\alpha$;
end while
if $F(proj(R_{z_k}(\alpha\eta_{z_k}))) < F(x_k)$ **then** ▷ Safeguard takes effect
    $x_k = R_{z_k}(\alpha\eta_{z_k})$, $y_k = R_{z_k}(\alpha\eta_{z_k})$, and $t_k = 1$;
    $x_k = proj(x_k)$, $y_k = proj(y_k)$;
else
    x_k, y_k and t_k keep unchanged;
end if
$z_{k+N} = x_k$; ▷ Update the compared iterate

where qr and svd mean computing the compact QR decomposition and SVD of a matrix, respectively. $R_X^{-1}(Y) = YS - X$, where S is the solution of the Lyapunov equation $(X^TY)S + S(Y^TX) = 2I_q$.

3.1.1 The Projection

Given $X \in St(q, n)$, the task is to find a $Y \in St(q, n)$ with $\mathbf{1}_n \in \mathcal{R}(Y)$ that minimizes $\|X - Y\|_F^2$. Letting $f(Y;\ X) = \operatorname{tr}(X^TY)$ denote a cost function parameterized by X, the problem can be formulated in an equivalent form by noting

$$\min_{Y \in \mathcal{F}} \|X - Y\|_F^2 \leftrightarrow \max_{Y \in \mathcal{F}} f(Y;\ X)$$

where $\mathcal{F} = \{Y \in St(q, n),\ \ \mathbf{1}_n \in \mathcal{R}(Y)\}$. The maximum value of $f(Y;\ X)$ is q and is achieved when $X \in \mathcal{F}$ and the problem is invariant with respect to $Q \in \mathcal{O}(q)$ where $\mathcal{O}(q)$ is the orthogonal group consisting of q-by-q orthogonal matrices, i.e.,

$$f(Y;\ X) = f(YQ;\ XQ).$$

Note that the cost function changes for this invariance. In general, $f(Y;\ X) \neq f(YQ;\ X)$.

For an element of the feasible set $\mathcal{F}$, there must exist $Q \in \mathcal{O}(q)$ such that $\hat{Y} = YQ = \begin{bmatrix} \tilde{1}_n & \tilde{Y} \end{bmatrix}$ with $\tilde{1}_n = \mathbf{1}_n/\sqrt{n}$, $\tilde{Y} \in St(q-1, n)$ and $\mathcal{R}(\tilde{Y}) \perp \tilde{1}_n$. There are, of course, many such $\hat{Y}$ possible. This can be seen from

$$Q = \begin{bmatrix} q_1 & Q_\perp \end{bmatrix}, \ \hat{Y} = \begin{bmatrix} \tilde{1}_n & \tilde{Y} \end{bmatrix} = \begin{bmatrix} Yq_1 & YQ_\perp \end{bmatrix}.$$

Given Y, the vector q_1 is uniquely defined but $Q_\perp$ is any orthonormal completion of q_1 and $\tilde{Y} = YQ_\perp$ varies with the choice of $Q_\perp$. This can be used to parameterize the cost function over $\mathcal{F}$ to give an alternative form of the optimization problem defining the projection and reveal a constructive form of the solution Y_*.

The two forms of the optimization problem are

$$Y_* = \underset{Y \in \mathcal{F}}{\operatorname{argmax}} \operatorname{tr}(X^T Y),$$
$$where\ \mathcal{F} = \{Y \in St(q,n),\ \ \mathbf{1}_n \in \mathcal{R}(Y)\};$$

$$\left(\hat{Y}_*, Q_*\right) = \underset{\hat{Y} \in \mathcal{G},\, Q \in \mathcal{O}(q)}{\operatorname{argmax}} \operatorname{tr}(Q^T X^T \hat{Y}),$$
$$\mathcal{G} = \{\hat{Y} = \begin{bmatrix} \tilde{\mathbf{1}}_n & \tilde{Y} \end{bmatrix} \mid \tilde{Y} \in St(q-1, n),\ \ \mathbf{1}_n \perp \mathcal{R}(\tilde{Y})\}.$$

The second form can be solved analytically and a solution for the first form recovered easily. The cost function for the second form can be expanded as

$$\operatorname{tr}(Q^T X^T \hat{Y}) = q_1^T X^T \tilde{\mathbf{1}}_n + \operatorname{tr}(Q_\perp^T X^T \tilde{Y}).$$

The first term of the sum in the cost function is independent of the second term while the second term is essentially determined by the choice of q_1. For any q_1 and orthonormal completion $Q_\perp$, the maximum value of q_1 for the second term is achieved by $\tilde{Y} = XQ_\perp$.

Given this optimal choice of $\tilde{Y}$ parameterized by Q, the problem then becomes finding the optimal q_* for

$$\max_{q_1 \in St(1,n)} \ q_1^T X^T \tilde{\mathbf{1}}_n$$

and $Q_* = \begin{bmatrix} q_* & Q_\perp^* \end{bmatrix}$ where $Q_\perp^*$ is any orthonormal completion of q_*. This has a maximum value of 1 if and only if $\tilde{\mathbf{1}}_n \in \mathcal{R}(X)$. Otherwise it is maximized by

$$q_* = \frac{X^T \tilde{\mathbf{1}}_n}{\|X^T \tilde{\mathbf{1}}_n\|_2}.$$

There are several maximizers given by

$$q_* = \frac{X^T \tilde{\mathbf{1}}_n}{\|X^T \tilde{\mathbf{1}}_n\|_2}$$
$$Q_\perp^* \in St(q-1, n) \text{ is any orthonormal completion of } q_*$$
$$\tilde{Y}_* = XQ_\perp^*,\ \hat{Y}_* = \begin{bmatrix} \tilde{\mathbf{1}}_n & \tilde{Y}_* \end{bmatrix}.$$

Finally, Y_*, the maximizer for the original parameterized form of $f(Y;\ X)$ can be determined from $\hat{Y}_*$

$$Y_* = \hat{Y}_* Q_*^T = \begin{bmatrix} \tilde{\mathbf{1}}_n & \tilde{Y}_* \end{bmatrix} \begin{bmatrix} q_* & Q_\perp^* \end{bmatrix}^T = \tilde{\mathbf{1}}_n q_*^T + XQ_\perp^*(Q_\perp^*)^T.$$

This form shows that the choice of $Q_\perp^*$, i.e., the basis for $\mathcal{R}^\perp(q_*)$, that determines $\hat{Y}_*$ does not result in multiple Y_* since the projector $Q_\perp^*(Q_\perp^*)^T$ is invariant.

Therefore, a computationally efficient form of the unique solution is given by

$$Y_* = \underset{Y \in \mathcal{F}}{\operatorname{argmax}}\, f(Y;\ X) = \tilde{\mathbf{1}}_n q_*^T + XQ_\perp^*(Q_\perp^*)^T \tag{3.3}$$
$$= \tilde{\mathbf{1}}_n q_*^T + X(I - q_* q_*^T), \tag{3.4}$$

where

$$q_* = \frac{X^T \tilde{\mathbf{1}}_n}{\|X^T \tilde{\mathbf{1}}_n\|_2}. \tag{3.5}$$

3.1.2 Semi-smooth Newton Method for the Proximal Subproblem

The remaining main part of Algorithm 3 is to solve the proximal subproblem efficiently. The proximal subproblem is as shown in (3.6):

$$V_k = \underset{V \in T_{X_k}\mathcal{F}_{\mathbf{1}_n}}{\operatorname{argmin}} \langle \operatorname{grad} f(X_k), V \rangle + \frac{1}{2\mu}||V||_F^2 + g(X_k + V). \tag{3.6}$$

Inspired by [25] and [53], we can apply the semi-smooth Newton method [119], [69] to solve (3.6). The general idea of semi-smooth Newton method is to solve a system of nonlinear equations based on the generalized Jacobian. So, we need to reduce the optimization problem to a system of nonlinear equations in order to use the semi-smooth Newton method. This can be obtained by considering the KKT conditions.

Define the linear operator $\mathcal{A}_k := X^T V + V^T X$. Therefore, we can rewrite the subproblem (3.6) as

$$V_k = \underset{\mathcal{A}_k(V)=0}{\operatorname{argmin}} \langle \operatorname{grad} f(X_k), V\rangle + \frac{1}{2\mu}||V||_F^2 + g(X_k + V). \tag{3.7}$$

By associating a Lagrange multiplier Λ to the linear equlity constraint, the Lagrangian function of (3.7) can be written as

$$\mathcal{L}(V;\Lambda) = \langle \operatorname{grad} f(X_k), V\rangle + \frac{1}{2\mu}||V||_F^2 + g(X_k + V) - \langle \mathcal{A}_k(V), \Lambda\rangle. \tag{3.8}$$

Then the KKT system of (3.7) is given by

$$\begin{cases} 0 \in \partial_V \mathcal{L}(V;\Lambda), \\ \mathcal{A}_k(V) = 0. \end{cases} \tag{3.9}$$

The first condition in (3.9) implies that V can be computed by

$$V(\Lambda) = prox_{\mu g}(B(\Lambda)) - X_k \text{ with } B(\Lambda) = X_k - \mu(\operatorname{grad} f(X_k) - \mathcal{A}_k^*(\Lambda)), \tag{3.10}$$

where

$$prox_{\mu g}(Z) = \underset{V\in\mathbb{R}^{n\times q}}{\operatorname{argmin}} \frac{1}{2}||V - Z||_F^2 + \mu g(V) \tag{3.11}$$

denotes the scaled proximal mapping, $\mathcal{A}^*$ denotes the adjoint of $\mathcal{A}$ and $\mathcal{A}^*(\Lambda) = X(\Lambda^T + \Lambda)$.

By substituting (3.10) into the second condition in (3.9), we can find that Λ satisfies

$$E(\Lambda) := \mathcal{A}_k(V(\Lambda)) = \mathcal{A}_k(prox_{\mu g}(X_k - \mu(\operatorname{grad} f(X_k) - \mathcal{A}^*(\Lambda))) - X_k) = 0, \tag{3.12}$$

which is a system of nonlinear equations with respect to Λ.

Therefore, to solve the proximal subproblem (3.6), we can find the solution to the nonlinear system (3.12) and then substitute it back to (3.10) to achieve V^*.

The nonlinear system (3.12) can be solved efficiently by the semi-smooth Newton method. Let Λ_k be the current estimate of the solution to (3.12). As in the Newton method, the key step in semi-smooth Newton method is to get a search direction by solving the following linear system

$$J_E(\Lambda_k)[d] = -E(\Lambda_k), \tag{3.13}$$

where $J_E(\Lambda_k)$ is the generalized Jacobian of E.

By the chain rule, we have

$$J_E(\Lambda_k)[d] = \mathcal{A}_k(\partial prox_{\mu g}(X_k - \mu(\text{grad}f(X_k) - \mathcal{A}^*(\Lambda_k)))) \circ (\mu\mathcal{A}^*(d)), \tag{3.14}$$

where $\partial prox_{\mu g}(\cdot)$ denotes the Clarke subdifferential [73] of $prox_{\mu g}(\cdot)$, and $\circ$ denotes the entrywise product of two matrices. Then we follow the framework of algorithm in [25].

3.2 ARPPG on Community Detection

As discussed in Section 2.1.3, there is a connection between Riemanian optimization and community detection. In this section, we describe more details on this for the ARPPG method.

3.2.1 Derivation of Global Maximum over Assignment Matrices

Firstly, the assignment matrices for community detection are defined.We denote a q dimensional vector with all entries being 1 by $\mathbf{1}_q$ and denote the $q \times q$ permutation matrices by P_q.

A matrix in the set of assignment matrices, $\mathcal{A}_{n,q}$, is defined as

Definition 3.2.1. *The matrix $X \in \{0,1\}^{n\times q}$, with $n \geq q$, is an assignment matrix if it satisfies*

(i) $X\mathbf{1}_q = \mathbf{1}_n$,

(ii) $X^TX = \mathbf{diag}(n_1, \cdots, n_q)$ *where* $n_i = ||Xe_i||_1$.

X is said to be in canonical ordering if the rows are permuted so that

$$X = \begin{pmatrix} \mathbf{1}_{n_1} & & & \\ & \mathbf{1}_{n_2} & & \\ & & \ddots & \\ & & & \mathbf{1}_{n_q} \end{pmatrix}.$$

Of course, the column ordering is not unique for the canonical form, i.e., XP_q is the same community assignment but with a different correspondence between the sets and the columns of the assignment matrix. For essential uniqueness, the additional constraint of $n_1 \geq n_2 \geq \ldots \geq n_q$ can be imposed. The columns are orthogonal, but not orthonormal, and X has exactly n nonzero elements all of which have the value of 1. As a result, X defines a partitioning of the indices $1, \ldots, n$ into q disjoint sets.

Now, let us look at the Modularity cost function. From [81], the scalar cost function $f(X)$ called modularity (up to a scalar $\frac{1}{2m}$) can be written as a quadratic function over $n \times q$ matrices defined by the matrix

$$M = A - \frac{A\mathbf{1}_n\mathbf{1}_n^T A}{2m}, \quad f(X) = \sum_{i,j}(A_{ij} - \frac{k_i k_j}{2m})\delta(\sigma_i, \sigma_j) = \mathrm{tr}(X^T M X),$$

where A is the adjacency matrix of the graph, M is the modularity matrix, m is the number of edges, n is the number of vertices in the graph and the total degree of the graph is $2m = \mathbf{1}_n^T A \mathbf{1}_n$.

The value of $f(X)$ is invariant under permutations on the columns of the assignment matrix X, i.e., $f(XP_q) = \mathrm{tr}(P_q^T X^T M X P_q)$. So there are multiple optimal ways of specifying the same community assignment.

Next, we look into the maximal of the modularity function on ideal graphs. An ideal graph is a graph where the communities are cliques and there are no edges between the cliques.

When A is an ideal graph with q communities we know it can be written [75]

$$A = \tilde{Z}_* \tilde{Z}_*^T$$

where $\tilde{Z}_* \in \mathcal{A}_{n,q}$ is not necessarily in canonical form and there exists a row permutation P so that

$$PAP^T = A_P = Z_* Z_*^T$$

where A_P is block diagonal with diagonal blocks $\mathbf{1}_{n_i}\mathbf{1}_{n_i}^T = z_i z_i^T$ for $1 \le i \le q$ and

$$Z_* = \begin{pmatrix} \mathbf{1}_{n_1} & & & \\ & \mathbf{1}_{n_2} & & \\ & & \ddots & \\ & & & \mathbf{1}_{n_q} \end{pmatrix} = \begin{pmatrix} z_1 & & & \\ & z_2 & & \\ & & \ddots & \\ & & & z_q \end{pmatrix}$$

is in canonical form.

The corresponding modularity matrices for an ideal A and the corresponding block diagonal A_P are given by

$$\begin{aligned} M = M^T &= A - \frac{A\mathbf{1}_n\mathbf{1}_n^T A}{2m} \\ &= \tilde{Z}_*\tilde{Z}_*^T - \frac{\tilde{Z}_*\tilde{Z}_*^T\mathbf{1}_n\mathbf{1}_n^T\tilde{Z}_*\tilde{Z}_*^T}{2m} \\ &= \tilde{Z}_*(I_q - \frac{\tilde{s}\tilde{s}^T}{2m})\tilde{Z}_*^T, \end{aligned}$$

and

$$M_P = Z_*(I_q - \frac{ss^T}{2m})Z_*^T,$$

where $s = Z_*^T\mathbf{1}_n = \begin{pmatrix} n_1 & \dots & n_q \end{pmatrix}^T$, and $2m = \mathbf{1}_n^T A_P \mathbf{1}_n = s^T s = \sum_{i=1}^q n_i^2$.

The cost function $f(X)$ is invariant under reorderings of A, so we can analyze any row ordering of Z_* denoted generically as Z below. The following result for the value of $f(Z)$, i.e., the cost function at the assignment matrix that generates the ideal matrix A, follows directly from the definitions.

Lemma 3.2.2. *If $A = ZZ^T$ for $Z \in \mathcal{A}_{n,q}$ then*

$$f(Z) = \sum_{i=1}^q n_i^2 - \frac{\sum_{i=1}^q n_i^4}{\sum_{i=1}^q n_i^2}.$$

We show that the value $f(X)$ for any $X \in \mathcal{A}_{n,q}$ is bounded above by $f(Z)$ in Theorem 3.2.5. The following lemmas are easily proven and are useful in proving the main result.

Lemma 3.2.3. *If $A = ZZ^T$ for $Z \in \mathcal{A}_{n,q}$ then for any $X \in \mathcal{A}_{n,q}$*

$$f(X) = \operatorname{tr}(X^T Z(I_q - \frac{ss^T}{2m})Z^T X) \leq \sum_{i=1}^q \gamma_i v_i^T X X^T v_i,$$

where $s = \begin{pmatrix} n_1 & \dots & n_q \end{pmatrix}^T$, $v_i = Ze_i$,

$$2m = \sum_{i=1}^q n_i^2,\ \gamma_i := 1 - \frac{n_i^2}{2m},$$

where $0 \leq \gamma_i < 1$.

Lemma 3.2.4. *Given $Z \in \mathcal{A}_{n,q}$, any $X \in \mathcal{A}_{n,q}$ satisfies*

$$v_i^T X X^T v_i \leq v_i^T Z Z^T v_i, \quad 1 \leq i \leq q$$

where $v_i = Ze_i$. Equality holds only when $X = ZP_q$, i.e., a column permutation of Z.

The desired result is stated as Theorem 3.2.5.

Theorem 3.2.5. *If $A = ZZ^T$ for $Z \in \mathcal{A}_{n,q}$ is an ideal adjacency matrix then for any $X \in \mathcal{A}_{n,q}$*

$$f(X) \leq f(Z),$$

where $f(X) = \operatorname{tr}(X^T Z(I_q - \frac{ss^T}{2m})Z^T X)$, $s = \begin{pmatrix} n_1 & \dots & n_q \end{pmatrix}^T$, $2m = \sum_{i=1}^q n_i^2$.

Proof. The series of lemmas above yields

$$f(X) \leq \sum_{i=1}^{q} \left(1 - \frac{n_i^2}{2m}\right) v_i^T Z Z^T v_i.$$

Note that

$$\begin{aligned}
&\sum_{i=1}^{q} \left(1 - \frac{n_i^2}{2m}\right) v_i^T Z Z^T v_i \\
&= \sum_{i=1}^{q} v_i^T Z Z^T v_i - \sum_{i=1}^{q} \frac{n_i^2}{2m} v_i^T Z Z^T v_i \\
&= \sum_{i=1}^{q} n_i^2 - \frac{\sum_{i=1}^{q} n_i^4}{2m} \\
&= \sum_{i=1}^{q} n_i^2 - \frac{\sum_{i=1}^{q} n_i^4}{\sum_{i=1}^{q} n_i^2} = f(Z).
\end{aligned}$$

□

Theorem 3.2.5 shows that the ideal graph assignment is a global maximum of the modularity function over $\mathcal{A}_{n,q}$.

3.2.2 Stiefel Manifold Algorithms for Community Detection

The algorithms discussed here assume the cost function

$$f(X) = \mathrm{tr}(X^T M X),$$

$$where \quad M = A - \frac{A\mathbf{1}_n \mathbf{1}_n^T A}{\mathbf{1}_n^T A \mathbf{1}_n}.$$

Now let us investigate the connection between the modularity matrix and the Stiefel manifold $St(q, n)$.

Lemma 3.2.6. *Let $Z \in \mathcal{A}_{n,q}$ and define $M = A - \frac{A\mathbf{1}\mathbf{1}^\intercal A}{\mathbf{1}^\intercal A \mathbf{1}}$. If A is the adjacency matrix of an ideal graph, then $A = ZZ^T$ and*

$$\mathcal{R}(A) = \mathcal{R}(Z) = (\mathcal{R}(M) \oplus^{\perp} \mathcal{R}(\mathbf{1}_n)), \tag{3.15}$$

$$\mathcal{N}(M) = (\mathcal{N}(Z^T) \oplus^{\perp} \mathcal{R}(\mathbf{1}_n)) = (\mathcal{N}(A) \oplus^{\perp} \mathcal{R}(\mathbf{1}_n)), \tag{3.16}$$

where $\mathcal{R}(A)$ denotes the range of A, $\mathcal{N}(A)$ denotes the null space of A, and $\oplus^{\perp}$ denotes the direct sum of two perpendicular spaces.

Proof. Note that the symmetry of A and M implies $\mathcal{N}(M) = \mathcal{R}(M)^{\perp}$ and $\mathcal{N}(A) = \mathcal{R}(A)^{\perp}$. Therefore (3.15) and (3.16) are equivalent. It follows from the definition of M that

$$\begin{aligned} M &= Z(I_q - \frac{ss^T}{s^Ts})Z^T = Z(I_q - \frac{ss^T}{s^Ts})^2 Z^T \\ &= [Z(I_q - \frac{ss^T}{s^Ts})][(I_q - \frac{ss^T}{s^Ts})Z^T]. \end{aligned}$$

This implies that

$$\mathcal{R}(M) = \mathcal{R}(Z(I_q - \frac{ss^T}{s^Ts})), \quad \mathcal{N}(M) = \mathcal{N}((I_q - \frac{ss^T}{s^Ts})Z^T),$$

and we also have

$$\mathcal{R}(A) = \mathcal{R}(Z), \quad \mathcal{N}(A) = \mathcal{N}(Z^T).$$

Since the projector $(I_q - \frac{ss^T}{s^Ts})$ has rank $n-1$, it follows that the ranges and null spaces of A and M have dimensions that can only differ by 1 at most. Now consider the vector $\mathbf{1}_n$. Since $s = Z^T\mathbf{1}_n$, we have $M\mathbf{1}_n = 0$ and since M is symmetric, $\mathbf{1}_n$ is orthogonal to $\mathcal{R}(M)$. Since $\mathbf{1}_n = Z\mathbf{1}_q$, we have $\mathbf{1}_n \in \mathcal{R}(Z) = \mathcal{R}(A)$. Together, these two properties prove (3.15), and hence also (3.16).

□

Since $rank(M) = q - 1$ and $M = M^T$, we have the eigendecomposition

$$M = X_* \Gamma X_*^T,$$

where

$$X_* \in St(q-1, n), \;\; \Gamma = diag(\gamma_1, \ldots, \gamma_{q-1}), \;\; \gamma_i \neq 0.$$

It then follows by Lemma 3.2.6 that $\left[X_* \;\; \frac{\mathbf{1}_n}{\sqrt{n}}\right] \in St(q, n)$ since $\mathcal{R}(X_*)$ is a subspace of $\mathcal{R}(M)$.

For the modularity matrix, the relationship between A and M is one of deflation of range that allows the characterization of the part of $\mathcal{R}(A) = \mathcal{R}(Z)$ that is removed when considering $\mathcal{R}(M)$ as shown in (3.15).

Therefore, we can now get the anticipated result of $\mathcal{R}(Z) = \mathcal{R}\left(\left[X_* \quad \frac{\mathbf{1}_n}{\sqrt{n}}\right]\right)$.

3.2.3 A Constrained Stiefel Optimization Problem

Multiple Extrema: Note that if a space $\mathcal{B}$ of dimension q has a basis that is an assignment matrix then it has $q!$ such bases all of which are of the form ZP_q where Z is any assignment matrix

basis and $P \in \{0,1\}^{q\times q}$ is a permutation matrix. All of these matrices have exactly n nonzero elements which is the minimum count possible for bases of the space. If the columns of such a matrix, Z, are normalized in Euclidean 2-norm length then an element of $St(q,n)$ is produced with n_i elements in column i all with the value $1/\sqrt{n_i}$ with $\sum_{i=1}^{q} n_i = n$. These are the global minima of

$$\min_{X\in St(q,n),\mathcal{R}(X)=\mathcal{B}} \|X\|_1,$$

where the l_1 norm is defined as $\|X\|_1 = \sum_{ij}\|X_{ij}\|$ imposing the sparsity of X.

In practical numerical computation, even on ideal matrices and certainly on problems for which noise perturbs A and Z from ideal, some projection is needed to take a matrix in $St(q,n)$ to the "nearest" matrix in $\mathcal{A}_{n,q}$.

A Constrained Stiefel Optimization Problem: The constrained Stiefel optimization problem used to perform community detection is

$$X_* = \operatorname*{argmax}_{X\in St(q,n),\ \mathbf{1}_n\in\mathcal{R}(X)} \operatorname{tr}(X^T M X) - \lambda\|X\|_1, \tag{3.17}$$

where $\lambda > 0$ is a tuning parameter controlling the balance between variance and sparsity. The approach to compute X_* is given in Algorithm 5.

Step 1 can be computed using any trace maximization algorithm. Our code uses RNewton in ROPTLIB [52].

Step 2 (ARPPG) is the main part of the algorithm, and it is inspired by [53]. In [53], the FISTA [9] is generalized from the Euclidean space to the Riemannian setting and the below general nonconvex optimization problem is considered

$$\min_{X\in\mathcal{M}} F(X) = f(X) + g(X), \tag{3.18}$$

where $\mathcal{M} \subset \mathbb{R}^{n\times q}$ is a Riemannian submanifold, $f : \mathbb{R}^{n\times q} \to \mathbb{R}$ is L-continuously differentiable (may be nonconvex) and g is continuous and convex but may not be differentiable.

The optimization problem (3.17) is a special case of problem (3.18), where $f(X) = \operatorname{tr}(X^T M X)$ is L-continuously differentiable and $g(X) = -\lambda\|X\|_1$ is continuous, convex, but not differentiable and the feasible set $\mathcal{F} \subset St(q,n)$ is a Riemannian submanifold. The details of ARPPG are in Algorithm 3.

In Step 3, we use the idea of continuation to choose the parameter λ that defines the cost function. We can get the optimal X_1^* after setting the initial λ_0 and X_0. We then increase λ_0

Algorithm 5 Algorithm for the Constrained Stiefel Optimization Problem

1: *Step 1:* Compute $Y_* \in St(q-1,n)$ where

$$Y_* = \underset{X \in St(q-1,n)}{\operatorname{argmax}} \operatorname{tr}(X^T M X).$$

and set the initial guess for Step 2 as

$$X_0 = \begin{bmatrix} Y_* & \frac{\mathbf{1}_n}{\sqrt{n}} \end{bmatrix}.$$

2: *Step 2 (ARPPG, see Algorithm 3):* Compute $X_* \in St(q,n), \mathbf{1}_n \in \mathcal{R}(X)$ where

$$X_* = \underset{X \in St(q,n),\, \mathbf{1}_n \in \mathcal{R}(X)}{\operatorname{argmax}} \operatorname{tr}(X^T M X) - \lambda \|X\|_1,$$

with X_0 as the initial guess.

3: *Step 3:* Get the assignment matrix $\hat{X}_*$ by setting the element with the largest magnitude in each row of X_* as 1, and the others as 0 when X_* is sufficiently sparse. Assess the assignment matrix $\hat{X}_*$ and determine whether it is acceptable as a solution to the community detection problem or if the parameter λ should be updated. If λ is updated then return to Step 2.

and use X_1^* as the initial matrix to get X_2^*. We continue this procedure until the cost function $\operatorname{tr}(X^T M X) - \lambda \|X\|_1$ does not improve anymore.

3.3 Numerical Experiments

In this section, we evaluate the empirical performance of ARPPG on community detection by comparing with some classical algorithms, the GN algorithm [84, 44], the Infomap method [98], and the Louvain Algorithm [12].

3.3.1 LFR Benchmark Model

The Lancichinetti–Fortunato–Radicchi benchmark [65] is an algorithm that generates benchmark networks (artificial networks that resemble real-world networks). They have a priori known communities and are used to compare different community detection methods. The advantage of this benchmark over others is that it accounts for the heterogeneity in the distributions of node degrees and of community sizes.

The node degrees and the community sizes are distributed according to a power law, with different exponents. The benchmark assumes that both the degree and the community size have

power law distributions with different exponents, τ_1 and τ_2 respectively. N is the number of nodes and the average degree is $< k >$. There is a mixing parameter $0 \leq \mu_{LFR} \leq 1$, which is the average fraction of neighboring nodes of a node that do not belong to any community that the benchmark node belongs to. This parameter controls the fraction of edges that are between communities. Thus, it reflects the amount of noise in the network. At the extremes, when $\mu_{LFR} = 0$ all links are within community links, if $\mu_{LFR} = 1$ all links are between nodes belonging to different communities.

One can generate the benchmark network using the following steps, and a software package to generate the benchmark graphs can be downloaded from `https://www.santofortunato.net/resources`.

Step 1: Generate a network with nodes following a power law distribution with exponent τ_1 and choose extremes of the distribution $k_{\min}$ and $k_{\max}$ to get desired average degree is $< k >$.

Step 2: $(1 - \mu_{LFR})$ fraction of links of every node is with nodes of the same community, while fraction μ_{LFR} is with the other nodes.

Step 3: Generate community sizes from a power law distribution with exponent τ_2 . The sum of all sizes must be equal to N. The minimal and maximal community sizes $s_{\min}$ and $s_{\max}$ must satisfy the definition of community so that every non-isolated node is in at least in one community: $s_{\min} > k_{\min}$, $s_{\max} > k_{\max}$.

Step 4: Initially, no nodes are assigned to communities. Then, each node is randomly assigned to a community. As long as the number of neighboring nodes within the community does not exceed the community size a new node is added to the community, otherwise stays out. In the following iterations the "homeless" node is randomly assigned to some community. If that community is complete, i.e. the size is exhausted, a randomly selected node of that community must be unlinked. Stop the iteration when all the communities are complete and all the nodes belong to at least one community.

Step 5: Implement rewiring of nodes keeping the same node degrees but only affecting the fraction of internal and external links such that the number of links outside the community for each node is approximately equal to the mixing parameter μ_{LFR}.

In the construction of the benchmark graphs, each node has a probability p_{in} of being connected to nodes in its group and a probability p_{out} of being connected to nodes in different groups. If

$p_{in} > p_{out}$, the groups are communities, otherwise, the network is essentially a random graph without community structure. A power law distribution is used.

The condition $p_{in} > p_{out}$ can be translated into a condition on the mixing parameter μ_{LFR}, which expresses the ratio between the external degree of a node with respect to its community and the total degree of the node [64]:

$$\mu_{LFR} = \frac{k_i^{out}}{k_i^{in} + k_i^{out}} < \frac{N - n_c}{N},$$

where k_i^{in} is the number of neighbors of node i that belong to its community c and k_i^{out} the number of neighbors of i that belong to the other communities, N is the number of nodes, n_c is the number of nodes of the community c.

Setting $\mu_{LFR} = 0$, gives a graph defining a ground truth where the communities are strongly connected components and there are no edges between the communities. This is more challenging than the ideal ground truth of communities that are cliques used to motivate the optimization problem. For any value of $\mu_{LFR} > 0$, the graph also has an associated ground truth but the mixing causes the community structure to be less clearly defined. For the LFR benchmarks, the networks have $N = 1000$ nodes, the average node degree is 20, the maximum node degree is 50, the communities have between 20 and 100 nodes, the exponent of the degree power law distribution is -2, and the exponent of the community size power law distribution is -1. The numbers of communities for the LFR benchmarks are around 20.

3.3.2 Real-World Networks

Three widely used real-world networks are used to assess the performance of ARPPG. The first is an American college football network [44], in which the nodes represent football teams, and an edge exists between the nodes if there is a match between two teams. The ground truth community assignment is given by the membership in the same athletic conference, i.e., indisputable observations. The second is Zachary's karate club network [123], which is an undirected social network of friendship between 34 members of a karate club at a university. Edges connect individuals who were observed to interact outside the activities of the karate club. The ground truth is based on the splitting of the membership into 2 new disjoint karate clubs. However, there is a second ground truth based of 4 communities of 2 disjoint social groups within each of the 2 new clubs. The 2 community ground truth is defined by indisputable observation, the 4 community ground truth

is based on less precise social interaction data. The third is the Polbooks network [82] of books about US politics published around the time of the 2004 presidential election and sold by the online bookseller Amazon.com. Edges between books represent frequent co-purchasing of books by the same buyers. The ground truth is determined by the subjective classification of the books by a non-expert human observer.

3.3.3 Comparing Partitions

Normalized Mutual Information (NMI): A ground truth assignment of nodes to communities is associated with each benchmark graph. Given the ground truth, normalized mutual information (NMI) [29] was used to compare the quality of the communities. NMI is a similarity measure between two partitions X and Y that represents their normalized mutual entropy and is defined

$$NMI(X,Y) = \frac{2\mathcal{I}(X,Y)}{\mathcal{H}(X)+\mathcal{H}(Y)},$$

where $\mathcal{H}(X)$ is the entropy of the partition X and $\mathcal{I}(X,Y)$ is the mutual information of the partitions X and Y given by

$$\mathcal{H}(X) = -\sum_{u} \frac{n_u}{N} \log \frac{n_u}{N},$$
$$\mathcal{I}(X,Y) = \sum_{u,v} \frac{n_{uv}}{N} \log \frac{N n_{uv}}{n_u n_v},$$

with n_u the number of nodes in community u and n_{uv} the number of common nodes in community u of partition X and community v of partition Y. The value of NMI is in $[0,1]$ with larger values indicating higher similarity.

The synthetic benchmarks have clearly defined ground truth based on intracommunity connectivity graphs that are strongly connected but not necessarily completely connected as in our ideal case defined above. The members of the family of networks are defined by a parameter that makes the network have an increasingly ill-defined community structure. As a result, any reasonable algorithm should detect community structure accurately when it is well-defined and the discrimination ability of the algorithm is tested as the definition degrades. Additionally, we must consider the robustness of the combinatorial algorithms relative to their runtime choices, e.g., the particular random walks used in Infomap or the order and manner in which one-node moves are considered in the Louvain method. Similarly, ARPPG and other algorithms based on optimization over a

continuous domain are dependent on their initial conditions or other strategies to avoid finding an unacceptable local maximum.

Adjusted Mutual Information (AMI):

- Mutual Information (MI)

  Given a set S of N elements $S = \{s_1, s_2, \ldots s_N\}$, consider two partitions of S, namely $U = \{U_1, U_2, \ldots, U_R\}$ with R clusters, and $V = \{V_1, V_2, \ldots, V_C\}$ with C clusters. It is presumed here that the partitions are so-called hard clusters; the partitions are pairwise disjoint:

  $$U_i \cap U_j = \varnothing = V_i \cap V_j,$$

  for all $i \neq j$, and complete:

  $$\cup_{i=1}^{R} U_i = \cup_{j=1}^{C} V_j = S.$$

  The mutual information of cluster overlap between U and V can be summarized in the form of an R$\times$ C contingency table $M = [n_{ij}]_{j=1\ldots C}^{i=1\ldots R}$, where n_{ij} denotes the number of objects that are common to clusters U_i and V_j. That is, $n_{ij} = |U_i \cap V_j|$.

  Suppose an object is picked at random from S; the probability that the object falls into cluster U_i is:

  $$P_U(i) = \frac{|U_i|}{N}.$$

  The entropy associated with the partitioning U is:

  $$H(U) = -\sum_{i=1}^{R} P_U(i) \log P_U(i).$$

  $H(U)$ is non-negative and takes the value 0 only when there is no uncertainty determining an object's cluster membership, i.e., when there is only one cluster. Similarly, the entropy of the clustering V can be calculated as:

  $$H(V) = -\sum_{j=1}^{C} P_V(j) \log P_V(j),$$

  where $P_V(j) = |V_j|/N$. The mutual information (MI) between two partitions:

  $$MI(U, V) = \sum_{i=1}^{R} \sum_{j=1}^{C} P_{UV}(i, j) \log \frac{P_{UV}(i, j)}{P_U(i) P_V(j)},$$

  where $P_{UV}(i, j)$ denotes the probability that a point belongs to both the cluster U_i in U and cluster V_j in V:

  $$P_{UV}(i, j) = \frac{|U_i \cap V_j|}{N}.$$

MI is a non-negative quantity upper bounded by the entropies $H(U)$ and $H(V)$. It quantifies the information shared by the two clusterings and thus can be employed as a clustering similarity measure.

- Adjustment for Chance (E(MI))

  Like the Rand index, the baseline value of mutual information between two random clusterings does not take on a constant value, and tends to be larger when the two partitions have a larger number of clusters (with a fixed number of set elements N). By adopting a hypergeometric model of randomness, it can be shown that the expected mutual information between two random clusterings is:

$$E\{MI(U,V)\} = \sum_{i=1}^{R}\sum_{j=1}^{C}\sum_{n_{ij}=(a_i+b_j-N)^+}^{\min(a_i,b_j)} \frac{n_{ij}}{N}\log\left(\frac{N\cdot n_{ij}}{a_i b_j}\right)\times \frac{a_i!b_j!(N-a_i)!(N-b_j)!}{N!n_{ij}!(a_i-n_{ij})!(b_j-n_{ij})!(N-a_i-b_j+n_{ij})!},$$

  where $(a_i + b_j - N)^+$ denotes $\max(1, a_i + b_j - N)$. The variables a_i and b_j are partial sums of the contingency table; that is, $a_i = \sum_{j=1}^{C} n_{ij}$ and $b_j = \sum_{i=1}^{R} n_{ij}$.

The adjusted measure for the mutual information [113] may then be defined to be:

$$AMI(U,V) = \frac{MI(U,V) - E\{MI(U,V)\}}{\max\{H(U),H(V)\} - E\{MI(U,V)\}}.$$

3.3.4 Numerical Results

Results for the LFR networks. For the LFR benchmark with $\mu_{LFR} = 0$, as expected and required, all four algorithms have $NMI = 1$, $AMI = 1$, the same modularity value and the same assignment to $q_{true} = 24$ strongly connected communities. AManPG requires the desired number of communities as a parameter value and in this case it was taken as $q = q_{true} = 24$. The choice of an initial q and the development of a dynamic adaptation strategy are key ongoing tasks for AManPG. There is promising evidence that it is possible. For $\mu_{LFR} = 0$ and AManPG run with $q = 25$ and $q = 26$, i.e., near q_{true}, the modularity decreases as q increases. The final values of NMI for $q = 25$ and $q = 26$ change only slightly 0.99 and 0.98 respectively. Of course this information is not available for the algorithm to use, but it is due to the fact that the partitioning for $q = 25$ and $q = 26$ are nested in the partitioning for $q = q_{true} = 24$, i.e., the extra communities are refinements of the 24 by splitting without crossing the ideal community boundaries. Any nodes that are not in the same

community in the ideal partitioning remain in different communities in the refined partitions. This information can be detected by the algorithm and used to guide adjustment of q while revealing a hierarchical structure relevant to discussion of resolution limits [41] and alternative cost functions, e.g., the constant Potts model [109]. The cost function for ARPPG in this dissertation is the configuration null model (CNM) without considering the resolution limits. A general framework of cost functions for community detection including the CNM with and without the resolution limits and constant Potts model is described in Appendix C.3.

The algorithms were also tested with multiple nonzero values of μ_{LFR}. The values of NMI, AMI and modularity are shown in Table 3.1.

Table 3.1: Performance on LFR Bechmark Networks

Methods	Measurements	The mixing parameter μ_{LFR}							
		0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
GN	NMI	1	1	1	0.9972	0.8694	0.6679	0.4932	0.4886
	AMI	1	1	1	0.9962	0.7202	0.2539	0.0142	0.0031
	Modularity	0.8254	0.7268	0.6283	0.5280	0.3579	0.1230	0.0393	0.0329
Infomap	NMI	1	1	1	1	1	0	0	0
	AMI	1	1	1	1	1	0	0	0
	Modularity	0.8254	0.7268	0.6283	0.5288	0.4440	0	0	0
Louvain	NMI	1	1	1	1	1	0.9527	0.2192	0.0677
	AMI	1	1	1	1	1	0.9107	0.1748	0.0267
	Modularity	0.8254	0.7268	0.6283	0.5288	0.4440	0.3390	0.2093	0.1921
ARPPG	NMI	1	1	1	1	0.9935	0.8811	0.3422	0.0967
	AMI	1	1	1	1	0.9927	0.8651	0.3014	0.0473
	Modularity	0.8254	0.7268	0.6283	0.5288	0.4427	0.3239	0.1712	0.1355

where ARPPG uses $q = q_{true}$ determined by the network for each value of μ_{LFR}.

All four methods determine the ground truth community assignments for the networks with $\mu_{LFR} \leq 0.3$. For $\mu_{LFR} = 0.4$ and $\mu_{LFR} = 0.5$ three methods determine the associated ground truths and one comes very close: GN with $NMI = 0.99$, $AMI = 0.99$ and ARPPG with $NMI = 0.99$, $AMI = 0.99$ respectively.

ARPPG using $q \neq q_{true}$ for $\mu_{LFR} \leq 0.4$ demonstrates trends like those for $\mu_{LFR} = 0$ upon which a q adaptation strategy might be built. As q increases from q_{true}, NMI, AMI and modularity decrease at a rate that increases as μ_{LFR} increases. The partitions are nested, then only slightly

not nested with one or two nodes crossing communities of the ground truth assignment, and finally with a significant loss of nesting.

For the noisy cases in Table 3.1, GN degrades quickly while ARPPG and the Louvain method degrade more slowly. Infomap achieves an $NMI = 1$, $AMI = 1$ until $\mu_{LFR} = 0.5$ then drops to near 0, because infomap can separate outliers and cut out them from some clusters and form their own cluster. The performance of Infomap and the Louvain method are sensitive to their runtime decisions, e.g., the Infomap performance here uses the heuristic available in the publicly distributed code of running the method multiple times and choosing the "best" result. ARPPG, on the other hand, with its continuation strategy and initial condition selection using RNewton was seen to be remarkably robust even in the noisy situations.

Results for Real-World Networks. In this section, we consider three widely used real-world networks are used for performance evaluation. The first is Zachary's karate club network [123], which is an undirected social network of friendship between 34 members of a karate club at a university. Edges connect individuals who were observed to interact outside the activities of the karate club. The second is the Polbooks network [82] of books about US politics published around the time of the 2004 presidential election and sold by the online bookseller Amazon.com. Edges between books represent frequent co-purchasing of books by the same buyers. The third is an American college football network [44], in which the nodes represent football teams, and an edge exists between the nodes if there is match between two teams. More games happen among teams within the same community than between teams from different communities.

We apply ARPPG algorithm to these three real-world networks and the results are shown in Table 3.2. From Table 3.2, we can find that ARPPG has better performance than the other algorithms on these three real world datasets, because it produces larger NMI than the others. ARPPG, therefore, detected more accurate community structures than the other methods. From the viewpoint of modularity, ARPPG produces the largest modularity of all the algorithms for Polbooks network, and produces slightly smaller modularities than Louvain algorithm for Zachary's karate club network and the American football network. However, the resulting numbers of communities by the Louvain algorithm are 4 and 10, respectively, which are inconsistent with ground truth for the two networks while the number of communities for ARPPG are the ground truth values of 2 and 12 respectively. For example, if we set the number of clusters q as 4 for karate network,

ARPPG produces the same modularity as the Louvain algorithm. In addition, the Louvain method is a greedy algorithm and does not allow for back-tracking, while ARPPG looks for any possible local improvement.

Table 3.2: Performance on Real-World Networks (the best performance is in bold), where n is the number of nodes, m is the number of edges, q_{true} is the number of ground truth communities and numbers in parentheses are the numbers of communities detected. For ARPPG the numbers in parentheses are also the values used for ARPPG's parameter q.

Datasets	n	m	q_{true}	Measurements	GN	Infomap	Louvain	ARPPG			
Football	115	613	12	NMI	0.879(10)	**0.924**(12)	0.890(10)	**0.924**(12)	0.911(13)	0.912(14)	0.882(10)
				AMI	0.802(10)	**0.898**(12)	0.821(10)	**0.898**(12)	0.861(13)	0.848(14)	0.813(10)
				Modularity	0.600(10)	0.601(12)	**0.605**(10)	0.601(12)	0.581(13)	0.566(14)	0.596(10)
Karate	34	78	2	NMI	0.580(5)	0.700(3)	0.587(4)	**1.000**(2)	0.811(3)	0.687(4)	0.542(5)
				AMI	0.402(5)	0.579(3)	0.425(4)	**1.000**(2)	0.672(3)	0.505(4)	0.364(5)
				Modularity	0.401(5)	0.402(3)	0.419(4)	0.372(2)	0.373(3)	**0.420**(4)	0.382(5)
Polbooks	105	441	3	NMI	0.559(5)	0.494(6)	0.537(5)	**0.565**(3)	0.503(4)	0.465(5)	0.439(6)
				AMI	0.488(5)	0.390(6)	0.458(5)	**0.535**(3)	0.424(4)	0.362(5)	0.323(6)
				Modularity	0.517(5)	0.523(6)	**0.527**(5)	0.508(3)	0.504(4)	0.510(5)	0.505(6)

Note that overall modularity values for the community assignments produced are significantly lower than those for the synthetic networks and the different assignments produced all have similar modularity values with significantly different quality as measured by NMI and AMI. This is most pronounced for the opinion-based ground truth of the Polbooks network as expected. For the football network, ARPPG using $q = q_{true}$ produces an assignment close to the ground truth. Infomap produces the same 12 community assignment but requires multiple runs, some of which produce significantly different assignments. GN and the Louvain method do not get the correct number of communities despite achieving a value of modularity close to that from the other algorithms. ARPPG run with $q \neq q_{true}$ exhibits the same trends on modularity and nesting discussed for the synthetic networks as desired.

For the karate club network, only ARPPG with $q = q_{true} = 2$ produces the ground truth with 2 communities. When ARPPG is run with $q \neq q_{true}$ it exhibits the desired nesting trends and, in particular, for $q = 4$ it produces the second ground truth known for the network. (The NMI and AMI in the table are not 1 for that case because it is the 4 community ground truth compared to the 2 community ground truth.) The Louvain algorithm produces four different community assignments depending on the order of traversal of the nodes. The 4 community ground truth is one of them but the one in the table are not quite the same as is seen from the NMI and AMI differing from that of

ARPPG. Infomap produces different community assignments with varying numbers of communities in different runs. The result in the table is the best one. As expected, the Polbooks network is the most difficult. Modularity does not predict well the quality of the assignment measured by NMI and AMI. Even ARPPG with $q = q_{true}$ does not produce an assignment as close to ground truth as it does for the other two networks. The fact that modularity does not clearly indicate the ground truth is also seen in the trends for ARPPG with $q \neq q_{true}$. Nesting is not observed and the best modularity is observed for $q = 5 \neq q_{true} = 3$.

3.4 Conclusion

In this chapter, we propose a new Riemannian projected proximal gradient method applied to modularity with a convex nonsmooth sparsity penalty term for community detection. Numerical results show that ARPPG is competetive with state-of-the-art algorithm in terms of quality assignment and robustness. The convergence of ARPPG is not guaranteed. If we can prove that the feasible set is an embedded submanifold of Stiefel manifold, then the AManPG in [53] can be used and the convergence theorem holds accordingly. The details are shown in Chapter 4. In addition, observations of ARPPG performance as algorithm parameters vary provide leading evidence that an efficient implementation are feasible. The recursive algorithm is proposed in Chapter 7.

CHAPTER 4

ACCELERATED MANIFOLD PROXIMAL GRADIENT OPTIMIZATION METHOD OVER THE FEASIBLE SET $\mathcal{F}_v$

From the theoretical and numerical results in Chapter 3, we can see that ARPPG is a reasonable algorithm for clustering problems, specifically for community detection. However, the convergence of ARPPG algorithm is not guaranteed. It is necessary to consider the geometry of the problem in more detail and to confirm that it admits the use of a Riemannian convergent algorithm. In [53], the accelerated Riemannian manifold proximal gradient algorithm (AManPG) is proposed and the global convergence for AManPG over the Riemanian manifold is shown. So, if we can prove that the feasible set $\mathcal{F}_{\mathbf{1}_n}$ is a Riemanian manifold and construct the manifold structure of the feasible set, then we can use AManPG directly over the feasible set $\mathcal{F}_{\mathbf{1}_n}$ with the confidence the corresponding convergence theorem implies. In this chapter, we show that the feasible set $\mathcal{F}_{\mathbf{1}_n}$ is an embedded submanifold of $\mathbb{R}^{n\times q}$ and the optimization-related Riemannian geometry is derived.

4.1 Manifold Structure of Feasible Set $\mathcal{F}_v$

In this section, we show that the feasible set $\mathcal{F}_{\mathbf{1}_n} = \{X \in St(q,n) : \mathbf{1}_n \in \mathcal{R}(X)\} = \{X \in \mathbb{R}^{n\times q} : X \in St(q,n) \text{ and } XX^T\mathbf{1}_n = \mathbf{1}_n\}$ is a submanifold of dimension $nq - q(q+1)/2 - n + q$ and derive the tangent space $T_X\mathcal{F}$, normal space $N_X\mathcal{F}$ and a practical retraction from $T_X\mathcal{F}_{\mathbf{1}_n}$ to $\mathcal{F}_{\mathbf{1}_n}$. Actually, we can show the more general feasible set $\mathcal{F}_v = \{X \in \mathbb{R}^{n\times q} : X^TX = I_q, v \in \text{span}(X)\}$, with $v > 0$ is an embedded submanifold of $\text{St}(q,n)$ in Theorem 4.1.1.

Theorem 4.1.1. *The set $\mathcal{F}_v$ is an embedded submanifold of $\text{St}(q,n)$ with dimension $\dim(\text{St}(q,n)) - (n-q) = nq - q(q+1)/2 - n + q$. Furthermore, $\mathcal{F}_v$ is an embedded submanifold of $\mathbb{R}^{n\times q}$ with the same dimension and $\mathcal{F}_v$ is compact.*

Proof. We verify that $\mathcal{F}_v$ is an embedded submanifold of $\text{St}(q,n)$ by following [13, Definition 8.70] using the notion of a local defining function. For any $X \in \mathcal{F}_v$, let $X_\perp$ be a matrix such that

$\begin{pmatrix} X & X_\perp \end{pmatrix}^T \begin{pmatrix} X & X_\perp \end{pmatrix} = I_n$. Therefore, by [35, (2.23)], we have that for any $V \in \mathrm{T}_X \operatorname{St}(q,n)$,

$$\operatorname{Exp}_X(V) = \begin{pmatrix} X & X_\perp \end{pmatrix} e^{\begin{pmatrix} \Omega & -K^T \\ K & 0 \end{pmatrix}} \begin{pmatrix} I_p \\ 0 \end{pmatrix},$$

defines the exponential mapping with respect to the canonical metric, where Exp denotes the matrix exponential, $V = X\Omega + X_\perp K$, and the canonical metric is $\langle \eta_X, \xi_X \rangle_X = \operatorname{tr}(\eta_X^T (I_n - \frac{1}{2} X X^T) \xi_X)$ for $\eta_X, \xi_X \in \mathrm{T}_X \operatorname{St}(q,n)$. By [34, Theorem 3.7], there exists a positive constant $\delta > 0$ such that Exp_X is a diffeomorphism in $\mathbb{B}(X,\delta)$. It follows that for any $Y \in \operatorname{Exp}_X(\mathbb{B}(X,\delta))$, the mapping $\operatorname{Exp}_X^{-1}(Y)$ is well-defined and $Y = \operatorname{Exp}_X(\operatorname{Exp}_X^{-1}(Y))$, i.e.,

$$Y = \begin{pmatrix} X & X_\perp \end{pmatrix} e^{\begin{pmatrix} X^T \operatorname{Exp}_X^{-1}(Y) & -\left(X_\perp^T \operatorname{Exp}_X^{-1}(Y)\right)^T \\ X_\perp^T \operatorname{Exp}_X^{-1}(Y) & 0 \end{pmatrix}} \begin{pmatrix} I_q \\ 0 \end{pmatrix}. \tag{4.1}$$

Define a function $\phi : \operatorname{Exp}_X(\mathbb{B}(x,\delta)) \to \mathbb{R}^{n \times (n-q)}$ by

$$\phi(Y) = \begin{pmatrix} X & X_\perp \end{pmatrix} e^{\begin{pmatrix} X^T \operatorname{Exp}_X^{-1}(Y) & -\left(X_\perp^T \operatorname{Exp}_X^{-1}(Y)\right)^T \\ X_\perp^T \operatorname{Exp}_X^{-1}(Y) & 0 \end{pmatrix}} \begin{pmatrix} 0 \\ I_{n-q} \end{pmatrix},$$

it follows from (4.1) that

$$\phi(Y)^T Y = 0_{(n-q)\times q}. \tag{4.2}$$

Since $\operatorname{Exp}_X^{-1}$ is smooth in $\operatorname{Exp}_X(\mathbb{B}(x,\delta))$, ϕ is a smooth function in its domain. Furthermore, it follows from $\phi(X)^T X_\perp = I_{n-q}$ that there exists a constant $\tilde{\delta} > 0$ such that $\phi(Z)^T X_\perp$ is full rank, i.e.,

$$\operatorname{rank}\left(\phi(Z)^T X_\perp\right) = n - q, \tag{4.3}$$

for any $Z \in \operatorname{Exp}_X(\mathbb{B}(x,\tilde{\delta}))$. Let $\hat{\delta} = \min(\delta, \tilde{\delta})$ and $\mathcal{N}_X = \operatorname{Exp}_X(\mathbb{B}(x,\hat{\delta}))$. We now define a function h by

$$h : \mathcal{N}_X \to \mathbb{R}^{n-q} : Y \mapsto h(Y) = \phi(Y)^T v.$$

Next, we verify that the function h is a local defining function in the sense that $h^{-1}(0) = \mathcal{N}_X \cap \mathcal{F}_v$ and $\mathrm{D}\, h(Y) : \mathrm{T}_Y \operatorname{St}(q,n) \to \mathbb{R}^{n-q}$ is surjective for any $Y \in \mathcal{N}_X$[1].

[1]Note that the local definition function only requires $\mathrm{D}\, h(Y)$ to be full rank at $Y = X$. Here, we prove a stronger result.

For any $Z \in h^{-1}(0)$, it holds that $\phi(Z)^T v = 0$ and $Z \in \mathrm{St}(q,n)$. Since $\phi(Z)^T v = 0$ implies $v \in \mathrm{span}(X)$, we have $Z \in \mathcal{F}_v$, which means $h^{-1}(0) \subseteq \mathcal{N}_X \cap \mathcal{F}_v$. On the other hand, for any $Z \in \mathcal{N}_X \cap \mathcal{F}_v$, it is obvious that $h(Z) = 0$, which means $\mathcal{N}_X \cap \mathcal{F}_v \subseteq h^{-1}(0)$. Overall, the equation $h^{-1}(0) = \mathcal{N}_X \cap \mathcal{F}_v$ holds.

Let V denote $\mathrm{Exp}_X^{-1}(Y)$. For any $U \in \mathrm{T}_{\mathrm{Exp}_X(-V)}\,\mathrm{St}(q,n)$, let $\dot{V} = \mathcal{T}^{-1}_{\mathrm{Exp}_{-V}}(-U)$ and $W = \mathcal{T}_{\mathrm{Exp}_V}\dot{V}$, where $\mathcal{T}_{\mathrm{Exp}}$ denotes the vector transport by differentiating the exponential mapping (4.1). Note that $\dot{V}$ is well-defined since $-V \in \mathbb{B}(X, \hat{\delta})$. We have

$$
\begin{aligned}
&\mathrm{D}\,h(Y)\,[W] = (\mathrm{D}\,\phi(Y)\,[W])^T\,v \\
&= \left(\begin{pmatrix} X & X_\perp \end{pmatrix} \mathrm{D}\,e^{\begin{pmatrix} X^T \mathrm{Exp}_X^{-1}(Y) & -\left(X_\perp^T \mathrm{Exp}_X^{-1}(Y)\right)^T \\ X_\perp^T \mathrm{Exp}_X^{-1}(Y) & 0 \end{pmatrix}} [W] \begin{pmatrix} 0 \\ I_{n-q} \end{pmatrix} \right)^T v \\
&= \begin{pmatrix} 0 & I_{n-q} \end{pmatrix} \left\{ \mathrm{D} \left(e^{\begin{pmatrix} -X^T \mathrm{Exp}_X^{-1}(Y) & \left(X_\perp^T \mathrm{Exp}_X^{-1}(Y)\right)^T \\ -X_\perp^T \mathrm{Exp}_X^{-1}(Y) & 0 \end{pmatrix}} \begin{pmatrix} I_p \\ 0 \end{pmatrix} \right) [W] \right\} \alpha,
\end{aligned}
$$

where $\alpha = X^T v$, $(e^A)^T = e^{-A}$ for any skew symmetric matrix A, and $X_\perp v = 0$.

Define the functions

$$
G(Y) = e^{\begin{pmatrix} -X^T \mathrm{Exp}_X^{-1}(Y) & \left(X_\perp^T \mathrm{Exp}_X^{-1}(Y)\right)^T \\ -X_\perp^T \mathrm{Exp}_X^{-1}(Y) & 0 \end{pmatrix}} \begin{pmatrix} I_p \\ 0 \end{pmatrix}, \text{ and}
$$

$$
H \begin{pmatrix} \Omega \\ K \end{pmatrix} = \mathrm{Exp}_X(X\Omega + X_\perp K).
$$

Since $V, \dot{V} \in \mathrm{T}_X\,\mathrm{St}(q,n)$, there exist skew symmetric matrices $\Omega_V, \Omega_{\dot{V}}$ and matrices $K_V, K_{\dot{V}}$ such that $V = X\Omega_V + X_\perp K_V$ and $\dot{V} = X\Omega_{\dot{V}} + X_\perp K_{\dot{V}}$. By the chain rule $\mathrm{D}\,G \circ H \begin{pmatrix} \Omega_V \\ K_V \end{pmatrix} \left[\begin{pmatrix} \Omega_{\dot{V}} \\ K_{\dot{V}} \end{pmatrix} \right] =$

$\mathrm{D}\, G\left(H\begin{pmatrix}\Omega_V\\ K_V\end{pmatrix}\right)\left[\mathrm{D}\, H\begin{pmatrix}\Omega_V\\ K_V\end{pmatrix}\left[\begin{pmatrix}\Omega_{\dot V}\\ K_{\dot V}\end{pmatrix}\right]\right]$ and $W = \mathcal{T}_{\mathrm{Exp}_V}\dot V$, we have that

$$\begin{aligned}
&\mathrm{D}\, e^{\begin{pmatrix}-X^T\operatorname{Exp}_X^{-1}(Y) & \left(X_\perp^T\operatorname{Exp}_X^{-1}(Y)\right)^T\\ -X_\perp^T\operatorname{Exp}_X^{-1}(Y) & 0\end{pmatrix}}\begin{pmatrix}I_p\\ 0\end{pmatrix}\left[\mathcal{T}_{R_V}\dot V\right]\\
=&\mathrm{D}\left(e^{\begin{pmatrix}-\Omega_V & K_V^T\\ -K_V & 0\end{pmatrix}}\begin{pmatrix}I_p\\ 0\end{pmatrix}\right)\left[\begin{pmatrix}\Omega_{\dot V}\\ K_{\dot V}\end{pmatrix}\right].
\end{aligned}$$

It follows that

$$\begin{aligned}
\mathrm{D}\, h(Y)\,[W] =& -\begin{pmatrix}0 & I_{n-q}\end{pmatrix}\mathrm{D}\left(e^{\begin{pmatrix}-\Omega_V & K_V^T\\ -K_V & 0\end{pmatrix}}\begin{pmatrix}I_p\\ 0\end{pmatrix}\right)\left[\begin{pmatrix}-\Omega_{\dot V}\\ -K_{\dot V}\end{pmatrix}\right]\alpha\\
=&\begin{pmatrix}0 & I_{n-q}\end{pmatrix}\begin{pmatrix}X & X_\perp\end{pmatrix}^T\mathcal{T}_{\mathrm{Exp}_{-V}}(\dot V) = \begin{pmatrix}0 & I_{n-q}\end{pmatrix}\begin{pmatrix}X & X_\perp\end{pmatrix}^T\mathcal{T}_{\mathrm{Exp}_{-V}}(\mathcal{T}^{-1}_{\mathrm{Exp}_{-V}}(-U))\\
=&-\begin{pmatrix}0 & I_{n-q}\end{pmatrix}\begin{pmatrix}X & X_\perp\end{pmatrix}^T U\alpha = -X_\perp^T U\alpha. \qquad (4.4)
\end{aligned}$$

Since U can be any tangent vector in $\mathrm{T}_{\mathrm{Exp}_X(-V)}\,\mathrm{St}(q,n)$ and $X_\perp^T\phi(\mathrm{Exp}_X(-V))$ is full rank by (4.3), the vector $-X_\perp^T U\alpha$ can be any one in $\mathbb{R}^{n-q}$. Therefore, $\mathrm{D}\, h(Y)$ is surjective and also full rank. Therefore, by [13, Definition 8.70], $\mathcal{F}_v$ is an embedded submanifold of $\mathrm{St}(q,n)$. Furthermore, by [13, Exercise 3.33], $\mathcal{F}_v$ is also an embedded submanifold of $\mathbb{R}^{n\times q}$.

Since $\mathcal{F}_v$ is a subset of $\mathrm{St}(q,n)$, it is a bounded set. Moreover, it is easy to show that its complement set in $\mathbb{R}^{n\times q}$ is an open set. Therefore, $\mathcal{F}_v$ is a closed set. It follows that $\mathcal{F}_v$ is compact. □

Theorem 4.1.2 gives the tangent space at any $X \in \mathcal{F}_v$ and its perpendicular space with respect to the Euclidean metric.

Theorem 4.1.2. *The tangent space of $\mathcal{F}_v$ at X is given by*

$$\mathrm{T}_X\,\mathcal{F}_v = \{X\Omega + X_\perp K : \Omega^T = -\Omega, K \in \mathbb{R}^{(n-q)\times q}, KX^Tv = 0\}$$

and the perpendicular space of $\mathrm{T}_X\,\mathcal{F}_v$, called the normal space at X, is given by

$$\mathrm{N}_X\,\mathcal{F}_v = \{XS + X_\perp uv^TX : S = S^T, u \in \mathbb{R}^{n-q}\}.$$

Proof. It follows from [13, Exercise 3.33] that $\mathrm{T}_X \mathcal{F}_v = \ker \mathrm{D}\, h(X)$. By (4.4), we have that for any $U \in \mathrm{T}_X \mathrm{St}(q,n)$, $\mathrm{D}\, h(X)[U] = -KX^T v$, where $U = X\Omega + X_\perp K$ and Ω is any skew symmetric matrix and K is any n-by-$(n-q)$ matrix. Therefore, it holds that $\ker \mathrm{D}\, h(X) = \{X\Omega + X_\perp K : \Omega^T = -\Omega, K \in \mathbb{R}^{(n-q)\times q}, KX^T v = 0\}$.

For any $V \in \mathrm{T}_X \mathcal{F}_v$ and $U \in N_X \mathcal{F}_v$, it is easy to verify that $\mathrm{trace}(U^T V) = 0$. In addition, $\dim(T_X \mathcal{F}_v) + \dim(\mathrm{N}_X \mathcal{F}_v) = nq - n - q - q(q+1)/2 + q(q+1)/2 + n + q = nq = \dim(\mathbb{R}^{n\times q})$. Therefore, $N_X \mathcal{F}_v = (\mathrm{T}_X \mathcal{F}_v)^\perp$ which implies $\mathrm{N}_X \mathcal{F}_v$ is the normal space of $\mathcal{F}_v$ at X. □

Theorem 4.1.3. *Given any $Z \in \mathbb{R}^{n\times q}$, the orthogonal projection to $\mathrm{N}_X \mathcal{F}_v$ is given by*

$$P_{\mathrm{N}_X}(Z) = X\frac{X^T Z + Z^T X}{2} + (I - XX^T)Z\hat{\alpha}\hat{\alpha}^T,$$

where $\hat{\alpha} = X^T v/\|X^T v\|$. The orthogonal projection to $\mathrm{T}_X \mathcal{F}_v$ is therefore

$$P_{\mathrm{T}_X}(Z) = X\frac{X^T Z - Z^T X}{2} + (I - XX^T)Z(I - \hat{\alpha}\hat{\alpha}^T).$$

Proof. By observing the formats of $P_{\mathrm{N}_X}(Z)$ and $P_{\mathrm{T}_X}(Z)$, we have $P_{\mathrm{N}_X}(Z) \in \mathrm{N}_X \mathcal{F}_v$ and $P_{\mathrm{T}_X}(Z) \in \mathrm{T}_X \mathcal{F}_V$. Therefore, the result follows from $P_{\mathrm{N}_X}(Z) + P_{\mathrm{T}_X}(Z) = XX^T Z + (I - XX^T)Z = Z$. □

Given $X \in \mathrm{St}(q,n)$, the orthonormal projection from X to $\mathcal{F}_v$ with $v = \mathbf{1}_n$ has been derived in [116]. The orthonormal projection with any $v > 0$ can be derived similarly. We state the result without proof in Lemma 4.1.4.

Lemma 4.1.4. *For any $X \in \mathrm{St}(q,n)$ with $X^T v \neq 0$, the global minimizer of the problem*

$$P_{\mathcal{F}_v}(X) = \operatorname*{argmin}_{Y \in \mathcal{F}_v} \|X - Y\|_F^2$$

is given by $Y_* = vq_*^T/\|v\|_2 + X(I - q_* q_*^T)$, where $q_* = X^T v/\|X^T v\|_2$.

One way to define a retraction on $\mathcal{F}_v$ is by the orthogonal projection [2], i.e.,

$$R_X^{\mathrm{proj}}(V) = P_{\mathcal{F}_v}(X + V), \tag{4.5}$$

where $X \in \mathcal{F}_v$ and $V \in \mathrm{T}_X \mathcal{F}_v$. However, we do not have a closed form solution of $P_{\mathcal{F}_v}(X + V)$ in general. A practical family of retractions is given in Theorem 4.1.5.

Theorem 4.1.5. *For any $X \in \mathcal{F}_v$, there exists a positive number $\delta_X > 0$ such that the mapping*

$$R_X : \mathbb{B}(x, \delta_X) \to \mathcal{F}_v : V \mapsto R_X(V) = P_{\mathcal{F}_v} \circ \tilde{R}_X \tag{4.6}$$

satisfies the two conditions of the retraction, i.e.,$R_X(0_X) = X$, where 0_X is the zero element in $T_X\mathcal{F}$; and $\frac{d}{dt}(R_X(tV))|_{t=0} = V$ for any $V \in \mathbb{B}(X, \delta_X)$, where $\tilde{R}$ is any retraction on $\mathrm{St}(q, n)$. Moreover, if the retraction $\tilde{R}$ satisfies $\mathrm{span}(\tilde{R}_X(V)) = \mathrm{span}(X + V)$, then the domain of $\tilde{R}$ in (4.6) is the whole tangent bundle. Such retractions $\tilde{R}$ include the retraction by qr decomposition in [1, (4.8)] and the retraction by polar decomposition in [1, (4.7)].

Proof. Since X satisfies $X^T v \neq 0$, $\tilde{R}_X(0_X) = X$, and $\tilde{R}_X$ is smooth, there exists a positive $\delta_X > 0$ such that $P_{\mathcal{F}_v}(\tilde{R}_X(V))$ is well-defined for any $V \in \mathbb{B}(X, \delta_X)$. The smoothness of R follows from the smoothness of $\tilde{R}$ and $P_{\mathcal{F}_v}$. We have $R_X(0_X) = P_{\mathcal{F}_v}(\tilde{R}_X(0_X)) = P_{\mathcal{F}_v}(X) = X$, where the second equality follows from the property of the retraction $\tilde{R}$ and the last equation follows from the definition of the projection $P_{\mathcal{F}_v}$.

In addition, we have

$$\begin{aligned}\frac{d}{dt}R_X(tV)|_{t=0} =& \frac{d}{dt}\left(P_{\mathcal{F}_v} \circ \tilde{R}_X\right)(tV)|_{t=0} = \left(\mathrm{D}\, P_{\mathcal{F}_v}(\tilde{R}_X(tV))\left[\frac{d}{dt}\tilde{R}_X(tV)\right]\right)|_{t=0} \\ =& \mathrm{D}\, P_{\mathcal{F}_v}(X)[V] = \mathrm{D}\, R_X^{\mathrm{proj}}(0_X)[V] = V,\end{aligned}$$

where the second equality follows from the chain rule, the third equality follows from $\tilde{R}_X(0_V) = X$ and $\frac{d}{dt}\tilde{R}_X(tV)|_{t=0} = V$, and the last equality follows from the fact that (4.5) is a retraction.

For the second part of the result, we only need to verify that $(X + V)^T v \neq 0$ for all $V \in \mathrm{T}_X\,\mathcal{F}_v$. Let $\alpha = X^T v \neq 0$. By the form of the tangent space $\mathrm{T}_X\,\mathcal{F}_v$ in Theorem 4.1.2, we have

$$\alpha^T (X + V)^T v = \alpha^T (X + X\Omega + X_\perp K)^T v = \alpha^T \alpha + \alpha^T \Omega \alpha = \|\alpha\|_2^2 \neq 0,$$

which implies $(X + V)^T v \neq 0$. □

By Theorem 4.1.5, two retractions of $\mathcal{F}_v$ based on the qr decomposition and the polar decomposition are respectively given by

$$R_X^{qf}(V) = v q_*^T / ||v||_2 + qf(X + V)(I - q_* q_*^T), \tag{4.7}$$

where $q_* = qf(X+V)^T v/||qf(X+V)^T v||_2$ and $qf(X+V)$ denotes the Q factor of the QR decomposition of $X+V$ that further satisfies the diagonal entries of the R factor being positive; and

$$R_X^{polar}(V) = vq_*^T/||v||_2 + (X+V)(I+V^TV)^{-1/2}(I-q_*q_*^T), \tag{4.8}$$

where $q_* = (I+V^TV)^{-1/2}(X+V)^T v/||(I+V^TV)^{-1/2}(X+V)^T v||_2$. As the dominant computations in (4.7) and (4.8) are the qr decomposition and the polar decomposition respectively, their computations both take $O(nq^2)$ flops. The retraction proposed in [20, (14)] is more computationally expensive. Specifically, the retraction in [20] is given by

$$R_X(V) = exp(B)exp(A')X, \tag{4.9}$$

where $A = X^TV, A' = XAX^T$, and $B = VX^T - XV^T - 2A'$. The computation of (4.9) requires an evaluation of an exponential of an n-by-n matrix B and therefore can be computationally unacceptable ($O(n^3)$ flops) when n is large.

4.2 AManPG Algorithm over $\mathcal{F}_v$

In [25], Chen et.al. considered a class of optimization problems over the Stiefel manifold whose objective function is the sum of a smooth, possibly nonconvex function and a convex, possibly nonsmooth function. The form of this class of optimization problems is as follows:

$$\min_{X\in\mathcal{M}} F(X) = f(X) + g(X), \tag{4.10}$$

where $\mathcal{M} = St(q,n) = \{X : X \in \mathbb{R}^{n\times q}, X^TX = I_q\}$, $f : \mathbb{R}^{n\times q} \to \mathbb{R}$ is L-continuously differentiable (may be nonconvex) and g is Lipschitz continuous and convex but may not be differentiable.

A manifold proximal gradient method named ManPG is proposed for this problem in [25], which is based on the proximal gradient method with a retraction operation to keep the iterates feasible with respect to the manifold constraint. ManPG is an analogue of the proximal gradient method in the Euclidean setting and hence can take advantage of the problem structure. Each step of ManPG involves solving a well-structured convex optimization problem which can be solved efficiently by the semi-smooth Newton method.

The AManPG algorithm proposed in [53] by extending the FISTA algorithm to solve (4.10). Empirical comparisons clearly show that AManPG has a faster convergence rate than ManPG as in the Euclidean case [53].

In this dissertation, we apply the AManPG algorithm [53] over the feasible set $\mathcal{F}_v$ to solve the problem (4.10), because we have already showed that the feasible set $\mathcal{F}_v = \{X \in \mathbb{R}^{n\times q} : X^TX = I_q, v \in \text{span}(X)\}$ is a Riemannian submanifold in Section 4.1. For simplicity of notation, throughout this section, we use lowercase x to denote a point in the domain manifold $\mathcal{M} := \mathcal{F}_v$ and use $g(x)$ to denote $\lambda\|x\|_1$. Therefore, the Riemanian optimization problem becomes

$$\min_{x\in\mathcal{M}} F(x) := f(x) + g(x). \tag{4.11}$$

The AManPG algorithm and convergence analysis rely on Assumptions 4.2.1. The convergence analysis in [53] of Theorem 3.1 can be applied here, because we have derived that $\mathcal{F}_v$ is a submanifold and $R_X(\cdot) = proj(\tilde{R}_X(\cdot))$ is a retraction from $T_X\mathcal{F}_v$ to $\mathcal{F}_v$. Note that $g(x) = \lambda\|x\|_1$ is a Lipschitz continuous function.

Assumption 4.2.1. *The gradient of f is Lipschitz continuous on $\mathcal{M}$ with a Lipschitz constant L_f and the function g is Lipschitz continuous with Lipschitz constant L_g, where the Lipschitz continuity is defined in the sense of the Euclidean setting.*

The accelerated manifold proximal gradient method proposed for Problem (4.11) is stated in [53, Algorithm 1] and [53, Algorithm 2]. They are included here for convenience as Algorithm 6 and Algorithm 7. We invite the reader to first read the more reader-friendly description in Section 3 of [53] and to refer to the pseudocode in Algorithm 6 when needed.

Algorithm 6 is a generalization of the accelerated proximal gradient method (FISTA) [9] to the embedded submanifold. The FISTA method consists of the following steps:

$$\begin{cases} \eta_{y_k} = \text{argmin}_{\eta\in\mathbb{R}^{n\times p}} \langle \nabla f(y_k), \eta\rangle + \frac{1}{2\mu}\|\eta\|_{\text{F}}^2 + g(y_k+\eta) \\ x_{k+1} = y_k + \eta_{y_k} \\ t_{k+1} = \frac{\sqrt{4t_k^2+1}+1}{2} \\ y_{k+1} = x_{k+1} + \frac{t_k-1}{t_{k+1}}(x_{k+1} - x_k), \end{cases} \tag{4.12}$$

where μ is a positive constant. The subproblem in the first step of (4.12) is generalized to the Riemannian setting by

$$\operatorname*{argmin}_{\eta\in \text{T}_{y_k}\mathcal{M}} \langle \text{grad}\, f(y_k), \eta\rangle + \frac{1}{2\mu}\|\eta\|^2 + g(y_k+\eta), \tag{4.13}$$

Algorithm 6 Accelerated Manifold Proximal Gradient Method (AManPG)

Require: Lipschitz constant L_f on ∇f, parameter $\mu > 0$ in the proximal mapping, line search parameter $\sigma \in (0, 1/(8\mu)]$, shrinking parameter in line search $\nu \in (0, 1)$, positive integer N for safeguard, initial iterates Λ_y and Λ_z for the semi-smooth Newton algorithm;

$t_0 = 1$, $y_0 = x_0$, $z_0 = x_0$;
for $k = 0, \ldots$ **do**
  if $\mod(k, N) = 0$ **then** ▷ Invoke safeguard every N iterations
    Invoke Algorithm 7: $[z_{k+N}, x_k, y_k, t_k, \Lambda_z] = Alg7(z_k, x_k, y_k, t_k, F(x_k), \Lambda_z)$;
  end if
  Compute $\eta_{y_k} = \operatorname{argmin}_{\eta \in \mathrm{T}_{y_k}\mathcal{M}} \langle \operatorname{grad} f(y_k), \eta \rangle + \frac{1}{2\mu}\|\eta\|^2 + g(y_k + \eta)$;
  $x_{k+1} = R_{y_k}(\eta_{y_k})$;
  $t_{k+1} = \frac{\sqrt{4t_k^2+1}+1}{2}$;
  Compute

$$y_{k+1} = R_{x_{k+1}}\left(\frac{1-t_k}{t_{k+1}} P_{T_{x_{k+1}}\mathcal{M}}(x_k - x_{k+1})\right);$$

end for

where η is required to be in the tangent space. Such generalization has been used in [25]. To solve the proximal subproblem 4.13, we can apply the semi-smooth Newton method [119], [69]. The general idea of semi-smooth Newton method is to solve a system of nonlinear equations based on the generalized Jacobian.

4.3 Basis of Normal Space of the Feasible Set $\mathcal{F}_v$ and Its Intrinsic Representation

As discussed in [53], to solve the proximal subproblem of the AManPG algorithm, we need to reduce the optimization problem to a system of nonlinear equations by considering the KKT conditions. One linear constraint $\mathcal{A}_k := N^T V = 0$ is introduced, where N is the basis of normal space (i.e., $(T_X\mathcal{F}_v)^\perp$) of the feasible set $\mathcal{F}_v$. If we can derive one orthonormal basis of the normal space which can be dealt with efficiently, then we can solve the proximal subproblem efficiently. The orthonormal basis of the noraml space of $\mathcal{F}_v$ is given in Section 4.3.1.

4.3.1 Orthonormal Basis of the Normal Space of the Feasible set $\mathcal{F}_v$

In this section, we construct an orthonormal basis $\mathcal{N}$ for the normal space $N_X\mathcal{F}_v$ to compute the following two tasks efficiently.

Algorithm 7 Safeguard for Algorithm 1

Require: $(z_k, x_k, y_k, t_k, F(x_k), \Lambda_z)$; The maximum number of iterations for line search $N_{\max} > 0$;

Ensure: $[z_{k+N}, x_k, y_k, t_k, \Lambda_z]$;

Compute $\eta_{z_k} = \operatorname{argmin}_{\eta \in \mathrm{T}_{z_k}\mathcal{M}} \langle \operatorname{grad} f(z_k), \eta \rangle + \frac{1}{2\mu}\|\eta\|^2_{W_{z_k}} + g(z_k + \eta)$;

Set $\alpha = 1$, and $i_{\text{iter}} = 0$;

while $F(R_{z_k}(\alpha\eta_{z_k})) > F(z_k) - \sigma\alpha\|\eta_{z_k}\|^2_{\mathrm{F}}$ and $i_{\text{iter}} < N_{\max}$ **do**

    $\alpha = \nu\alpha$; $i_{\text{iter}} = i_{\text{iter}} + 1$;

end while

if $i_{\text{iter}} = N_{\max}$ **then**

    Line search fails;

end if

if $F(R_{z_k}(\alpha\eta_{z_k})) < F(x_k)$ **then** ▷ Safeguard takes effect

    $x_k = R_{z_k}(\alpha\eta_{z_k})$, $y_k = R_{z_k}(\alpha\eta_{z_k})$, and $t_k = 1$;

else

    x_k, y_k and t_k keep unchanged;

end if

$z_{k+N} = x_k$; ▷ Update the compared iterate;

(i) Given an element in $N_X\mathcal{F}_v$, get the coordinates under the basis $\mathcal{N}$;

(ii) Given the coordinates, get the element in $N_X\mathcal{F}_v$ under the basis $\mathcal{N}$.

If $N \in N_X\mathcal{F}_v$, then

$$N = X \begin{bmatrix} c_{11} & c_{12} & c_{13} & \cdots & c_{1q} \\ c_{12} & c_{22} & c_{23} & \cdots & c_{2q} \\ c_{13} & c_{23} & c_{33} & \cdots & c_{3q} \\ \vdots & \vdots & \vdots & \ddots & \cdots \\ c_{1q} & c_{2q} & c_{3q} & \cdots & c_{qq} \end{bmatrix} + X_\perp \begin{bmatrix} k_1 \\ \vdots \\ k_{n-q} \end{bmatrix} v^T X.$$

So, one way to define the basis of $N_X\mathcal{F}$ is that

$$\{X(\mathbf{e}_i\mathbf{e}_j^T + \mathbf{e}_j\mathbf{e}_i^T) : i = 1, \cdots, q, j = 1, \cdots, q\} \cup \{X_\perp \tilde{\mathbf{e}}_i \tilde{v}^T X, i = 1, \cdots, n-q\}, \tag{4.14}$$

where $(\mathbf{e}_1, \cdots, \mathbf{e}_q)$ is the canonical basis of $\mathbb{R}^q$, $(\tilde{\mathbf{e}}_1, \cdots, \tilde{\mathbf{e}}_{n-q})$ is the canonical basis of $\mathbb{R}^{n-q}$ and $\tilde{v} = v/||v||$ is the normalized vector of v. Then the corresponding coordinates of N are $(c_{11}/2, c_{12}, \cdots, c_{1q}, c_{22}/2, \cdots, c_{2q}, \cdots, c_{qq}/2, k_1, k_2, \cdots, k_{n-q})$ under this basis.

Proposition 4.3.1 (Orthonormal Basis I of $N_X\mathcal{F}$)**.** *The basis defined in* (4.14) *is an orthonormal basis with respect to the canonical metric*

$$g_c(Z_1, Z_2) = \operatorname{tr}(Z_1^T(I_n - \frac{1}{2}XX^T)Z_2), \tag{4.15}$$

where $Z_1, Z_2 \in N_X\mathcal{F}$.

Proof. Case (i): $U = X(\mathbf{e}_i\mathbf{e}_j^T + \mathbf{e}_j\mathbf{e}_i^T)$

$$\begin{aligned}
g_c(U,U) &= \operatorname{tr}(U^TU - \frac{1}{2}U^TXX^TU) \\
&= \operatorname{tr}((\mathbf{e}_j\mathbf{e}_i^T + \mathbf{e}_i\mathbf{e}_j^T)X^TX(\mathbf{e}_i\mathbf{e}_j^T + \mathbf{e}_j\mathbf{e}_i^T) - \frac{1}{2}(\mathbf{e}_j\mathbf{e}_i^T + \mathbf{e}_i\mathbf{e}_j^T)X^TXX^TX(\mathbf{e}_i\mathbf{e}_j^T + \mathbf{e}_j\mathbf{e}_i^T) \\
&= \frac{1}{2}\operatorname{tr}((\mathbf{e}_j\mathbf{e}_i^T + \mathbf{e}_i\mathbf{e}_j^T)(\mathbf{e}_i\mathbf{e}_j^T + \mathbf{e}_j\mathbf{e}_i^T) \\
&= \frac{1}{2}\operatorname{tr}(\mathbf{e}_j\mathbf{e}_i^T\mathbf{e}_i\mathbf{e}_j^T + \mathbf{e}_j\mathbf{e}_i^T\mathbf{e}_j\mathbf{e}_i^T + \mathbf{e}_i\mathbf{e}_j^T\mathbf{e}_i\mathbf{e}_j^T + \mathbf{e}_i\mathbf{e}_j^T\mathbf{e}_j\mathbf{e}_i^T) \\
&= \frac{1}{2}\operatorname{tr}(\mathbf{e}_j^T\mathbf{e}_j + \mathbf{e}_i^T\mathbf{e}_i) \\
&= 1.
\end{aligned}$$

Case (ii): $U = X(\mathbf{e}_i\mathbf{e}_j^T + \mathbf{e}_j\mathbf{e}_i^T), V = X(\mathbf{e}_i\mathbf{e}_k^T + \mathbf{e}_k\mathbf{e}_i^T), j \neq k$

$$\begin{aligned}
g_c(U,V) &= \operatorname{tr}(U^TV - \frac{1}{2}U^TXX^TV) \\
&= \operatorname{tr}((\mathbf{e}_j\mathbf{e}_i^T + \mathbf{e}_i\mathbf{e}_j^T)X^TX(\mathbf{e}_i\mathbf{e}_k^T + \mathbf{e}_k\mathbf{e}_i^T) - \frac{1}{2}(\mathbf{e}_j\mathbf{e}_i^T + \mathbf{e}_i\mathbf{e}_k^T)X^TXX^TX(\mathbf{e}_i\mathbf{e}_k^T + \mathbf{e}_k\mathbf{e}_i^T) \\
&= \frac{1}{2}\operatorname{tr}((\mathbf{e}_j\mathbf{e}_i^T + \mathbf{e}_i\mathbf{e}_j^T)(\mathbf{e}_i\mathbf{e}_k^T + \mathbf{e}_k\mathbf{e}_i^T)) \\
&= \frac{1}{2}\operatorname{tr}(\mathbf{e}_j\mathbf{e}_i^T\mathbf{e}_i\mathbf{e}_k^T + \mathbf{e}_j\mathbf{e}_i^T\mathbf{e}_k\mathbf{e}_i^T + \mathbf{e}_i\mathbf{e}_j^T\mathbf{e}_i\mathbf{e}_k^T + \mathbf{e}_i\mathbf{e}_j^T\mathbf{e}_k\mathbf{e}_i^T) \\
&= 0.
\end{aligned}$$

Case (iii): $U = X(\mathbf{e}_i\mathbf{e}_j^T + \mathbf{e}_j\mathbf{e}_i^T), V = X(\mathbf{e}_k\mathbf{e}_j^T + \mathbf{e}_j\mathbf{e}_k^T), i \neq k$

$$g_c(U,V) = \frac{1}{2}\operatorname{tr}((\mathbf{e}_j\mathbf{e}_i^T + \mathbf{e}_i\mathbf{e}_j^T)(\mathbf{e}_k\mathbf{e}_j^T + \mathbf{e}_j\mathbf{e}_k^T)) = 0$$

Case (iv): $U = X(\mathbf{e}_i\mathbf{e}_j^T + \mathbf{e}_j\mathbf{e}_i^T), V = X(\mathbf{e}_k\mathbf{e}_l^T + \mathbf{e}_l\mathbf{e}_k^T), i \neq k, j \neq l$

$$g_c(U,V) = \frac{1}{2}\operatorname{tr}((\mathbf{e}_j\mathbf{e}_i^T + \mathbf{e}_i\mathbf{e}_j^T)(\mathbf{e}_k\mathbf{e}_l^T + \mathbf{e}_l\mathbf{e}_k^T)) = 0$$

Case (v): $U = X(\mathbf{e}_i\mathbf{e}_j^T + \mathbf{e}_j\mathbf{e}_i^T), V = X_\perp\tilde{\mathbf{e}}_i\tilde{v}^TX$

$$g_c(U,V) = \operatorname{tr}((\mathbf{e}_j\mathbf{e}_i^T + \mathbf{e}_i\mathbf{e}_j^T)X^TX_\perp\tilde{\mathbf{e}}_i\tilde{v}^TX - \frac{1}{2}(\mathbf{e}_j\mathbf{e}_i^T + \mathbf{e}_i\mathbf{e}_i^T)X^TXX^TX_\perp\tilde{\mathbf{e}}_i\tilde{v}^TX) = 0$$

Case (vi): $U = X_\perp \tilde{\mathbf{e}}_i \tilde{v}^T X$

$$\begin{aligned} g_c(U,U) &= \mathrm{tr}(X^T \tilde{v} \tilde{\mathbf{e}}_i^T X_\perp^T X_\perp \tilde{\mathbf{e}}_i \tilde{v}^T X - \frac{1}{2} X^T \tilde{v} \tilde{\mathbf{e}}_i X_\perp^T X X^T X_\perp \tilde{\mathbf{e}}_i \tilde{v}^T X) \\ &= \mathrm{tr}(X^T \tilde{v} \tilde{\mathbf{e}}_i^T \tilde{\mathbf{e}}_i \tilde{v}^T X) = \mathrm{tr}(\tilde{v}^T X X^T \tilde{v}) = \mathrm{tr}(\tilde{v}^T \tilde{v}) = 1. \end{aligned}$$

Case (vii): $U = X_\perp \tilde{\mathbf{e}}_i \tilde{v}^T X$, $V = X_\perp \tilde{\mathbf{e}}_j \tilde{v}^T X$, $i \neq j$

$$\begin{aligned} g_c(U,V) &= \mathrm{tr}(X^T \tilde{v} \tilde{\mathbf{e}}_i^T X_\perp^T X_\perp \tilde{\mathbf{e}}_j \tilde{v}^T X - \frac{1}{2} X^T \tilde{v} \tilde{\mathbf{e}}_i X_\perp^T X X^T X_\perp \tilde{\mathbf{e}}_j \tilde{v}^T X) \\ &= \mathrm{tr}(X^T \tilde{v} \tilde{\mathbf{e}}_i^T \tilde{\mathbf{e}}_j \tilde{v}^T X) = 0. \end{aligned}$$

□

If we consider the embedding metric $g_e(Z_1, Z_2) = \mathrm{tr}(Z_1^T Z_2)$, then we can just do a simple modification of (4.14) as follows:

$$\mathcal{B}_X = \{X \mathbf{e}_i \mathbf{e}_i^T, i = 1, ..., q\} \cup \{\frac{1}{\sqrt{2}} X(\mathbf{e}_i \mathbf{e}_j^T + \mathbf{e}_j \mathbf{e}_i^T) : i = 1, \cdots, q, j = i+1, \cdots, q\} \cup \{X_\perp \tilde{\mathbf{e}}_i \tilde{v}^T X, i = 1, \cdots, n-q\}, \tag{4.16}$$

where $(\mathbf{e}_1, \cdots, \mathbf{e}_q)$ is the canonical basis of $\mathbb{R}^q$, $(\tilde{\mathbf{e}}_1, \cdots, \tilde{\mathbf{e}}_{n-q})$ is the canonical basis of $\mathbb{R}^{n-q}$ and $\tilde{v} = v/||v||$ is the normalized vector of$v$. Then the corresponding coordinates of N are $(c_{11}, \sqrt{2}c_{12}, \cdots, \sqrt{2}c_{1q}, c_{22}, \cdots, \sqrt{2}c_{2q}, \cdots, c_{qq}, k_1, k_2, \cdots, k_{n-q})$ under this basis.

Proposition 4.3.2 (Orthonormal Basis II of $N_X\mathcal{F}$)**.** *The basis defined in* (4.16) *is an orthonormal basis with respect to the canonical metric*

$$g_e(Z_1, Z_2) = \mathrm{tr}(Z_1^T Z_2), \tag{4.17}$$

where $Z_1, Z_2 \in N_X\mathcal{F}$.

Proof. Case (i): $U = X \mathbf{e}_i \mathbf{e}_i^T$

$$g_e(U,U) = \mathrm{tr}(\mathbf{e}_i \mathbf{e}_i^T X^T X \mathbf{e}_i \mathbf{e}_i^T) = \mathrm{tr}(\mathbf{e}_i \mathbf{e}_i^T) = 1.$$

Case (ii): $U = \frac{1}{\sqrt{2}} X(\mathbf{e}_i \mathbf{e}_j^T + \mathbf{e}_j \mathbf{e}_i^T), i \neq j$

$$\begin{aligned} g_e(U,U) &= \mathrm{tr}(\frac{1}{\sqrt{2}}(\mathbf{e}_j \mathbf{e}_i^T + \mathbf{e}_i \mathbf{e}_j^T) X^T \frac{1}{\sqrt{2}} X(\mathbf{e}_i \mathbf{e}_j^T + \mathbf{e}_j \mathbf{e}_i^T)) \\ &= \mathrm{tr}(\frac{1}{2}(\mathbf{e}_j \mathbf{e}_i^T \mathbf{e}_i \mathbf{e}_j^T + \mathbf{e}_j \mathbf{e}_i^T \mathbf{e}_j \mathbf{e}_i^T + {}_i \mathbf{e}_j^T \mathbf{e}_i \mathbf{e}_j^T + \mathbf{e}_i \mathbf{e}_j^T \mathbf{e}_j \mathbf{e}_i^T)) \\ &= \frac{1}{2} \mathrm{tr}(\mathbf{e}_j \mathbf{e}_j^T + \mathbf{e}_i \mathbf{e}_i^T) = 1. \end{aligned}$$

Case (iii): $U = X\mathbf{e}_i\mathbf{e}_i^T, V = X\mathbf{e}_j\mathbf{e}_j^T, i \neq j$

$$g_e(U,V) = \mathrm{tr}(\mathbf{e}_i\mathbf{e}_i^T X^T X\mathbf{e}_j\mathbf{e}_j^T) = 0.$$

Case (iv): $U = X\mathbf{e}_i\mathbf{e}_i^T, V = \frac{1}{\sqrt{2}}X(\mathbf{e}_j\mathbf{e}_k^T + \mathbf{e}_k\mathbf{e}_j^T), j \neq k$

$$g_e(U,V) = \mathrm{tr}(\mathbf{e}_i\mathbf{e}_i^T X^T \frac{1}{\sqrt{2}}X(\mathbf{e}_j\mathbf{e}_k^T + \mathbf{e}_k\mathbf{e}_j^T)) = 0$$

Case (iv): $U = \frac{1}{\sqrt{2}}X(\mathbf{e}_i\mathbf{e}_j^T + \mathbf{e}_j\mathbf{e}_i^T), V = \frac{1}{\sqrt{2}}X(\mathbf{e}_k\mathbf{e}_l^T + \mathbf{e}_l\mathbf{e}_k^T), i \neq j, and\ k \neq l, and$ "$if\ i = k, then\ j \neq l$" *or* "$if\ j = l,\ i \neq k$" It is easy to verify

$$g_e(U,V) = 0.$$

Case (v): $U = X\mathbf{e}_i\mathbf{e}_i^T, V = X_\perp\tilde{\mathbf{e}}_k\tilde{v}^T X$

$$g_e(U,V) = \mathrm{tr}(\mathbf{e}_i\mathbf{e}_i^T X^T X_\perp\tilde{\mathbf{e}}_k\tilde{v}^T X) = 0.$$

Case (vi): $U = \frac{1}{\sqrt{2}}X(\mathbf{e}_i\mathbf{e}_j^T + \mathbf{e}_j\mathbf{e}_i^T), V = X_\perp\tilde{\mathbf{e}}_k\tilde{v}^T X$

$$g_e(U,V) = \mathrm{tr}(\frac{1}{\sqrt{2}}(\mathbf{e}_j\mathbf{e}_i^T + \mathbf{e}_i\mathbf{e}_j^T)X^T X_\perp\tilde{\mathbf{e}}_k\tilde{v}^T X) = 0.$$

Case (vii): $U = X_\perp\tilde{\mathbf{e}}_i\tilde{v}^T X$

$$g_e(U,U) = \mathrm{tr}(X^T\tilde{v}\tilde{\mathbf{e}}_i^T X_\perp^T X_\perp\tilde{\mathbf{e}}_i\tilde{v}^T X) = \mathrm{tr}(\tilde{\mathbf{e}}_i^T\tilde{\mathbf{e}}_i\tilde{v}^T XX^T\tilde{v}) = \mathrm{tr}(\tilde{\mathbf{e}}_i^T\tilde{\mathbf{e}}_i) = 1$$

Case(viii): $U = X_\perp\tilde{\mathbf{e}}_i\tilde{v}^T X, V = X_\perp\tilde{\mathbf{e}}_j\tilde{v}^T X, i \neq j$

$$g_e(U,V) = \mathrm{tr}(X^T\tilde{v}\tilde{\mathbf{e}}_i^T X_\perp^T X_\perp\tilde{\mathbf{e}}_j\tilde{v}^T X) = \mathrm{tr}(\tilde{\mathbf{e}}_i^T\tilde{\mathbf{e}}_j\tilde{v}^T XX^T\tilde{v}) = \mathrm{tr}(\tilde{\mathbf{e}}_i^T\tilde{\mathbf{e}}_j) = 0.$$

□

We use the orthonomal basis II in (4.16) for the AManPG algorithm and the inexact AManPG algorithm in this dissertation. Let V_i, $i = 1, \ldots q(q+1)/2 + n - q$ denote the entries in the basis $\mathcal{B}_X$. Define a function by

$$B_X : \mathbb{R}^{q(q+1)/2-n-q} \to \mathbb{R}^{n\times q} : u \to B_X u = \sum_{i=1}^{q(q+1)/2+n-q} u_i V_i \in N_x\mathcal{F}_v$$

and another function by

$$B_X^T : \mathbb{R}^{n\times p} \to \mathbb{R}^{q(q+1)/2-n-q} : V \to u,$$

where $u_i = \mathrm{trace}(V^T V_i)$. These two functions are used in the AManPG method given in Section 4.2 and the inexact AManPG method in Section 5.1.

4.3.2 Intrinsic Representation

Though the matrix $X_\perp$ is used in the orthonomal basis in (4.16), we do not need to construct such a matrix and only need to use two mappings $\alpha_X : \mathbb{R}^{n\times q} \to \mathbb{R}^{n\times q} : A \mapsto [XX_\perp]^T A$ and $\beta : \mathbb{R}^{n\times q} \to \mathbb{R}^{n\times q} : A \mapsto [XX_\perp]A$ by using the intrinsic representation. These two mappings can be computed efficiently in $O(nq^2)$, see details in [51].

The intrinsic representation of $N \in N_X\mathcal{F}_v$, which is given by

$$E2D_X^{\mathcal{F}_v}(N) = (\sqrt{2}c_{11}/2, \sqrt{2}c_{12}, \cdots, \sqrt{2}c_{1q}, \sqrt{2}c_{22}/2, \cdots, \sqrt{2}c_{2q}, \cdots, \sqrt{2}c_{qq}/2, k_1, k_2, \cdots, k_{n-q}). \tag{4.18}$$

By using the intrinsic representation, we can compute the following efficiently.

(i) Given an element in $N_X\mathcal{F}_v$, N, to compute the coordinates v_X under the basis $\mathcal{B}_X$, see Algorithm 8 for the details;

(ii) Given the coordinates, v_X, to compute the element N in $N_X\mathcal{F}$ under the basis $\mathcal{B}_X$, see Algorithm 9 for the details.

Algorithm 8 Computation of $E2D_X^{\mathcal{F}_v}(N)$

Require: $X \in \mathcal{F}_v, N \in N_X\mathcal{F}_v$, a positive vector $v \in \mathbb{R}^n$, a function $\alpha_X : \mathbb{R}^{n\times q} \to \mathbb{R}^{n\times q} : A \mapsto [XX_\perp]^T A$ generated by Algorithm 11;

1: $\begin{bmatrix}\Omega \\ K\end{bmatrix} = \alpha_X(N)$, where $\Omega \in \mathbb{R}^{q\times q}, K \in \mathbb{R}^{(n-q)\times q}$;
2: Set $\Omega = (\Omega + \Omega^T)/2$ and $k = 1$;
3: $\Omega_{ii} = \Omega_{ii}/2$;
4: **for** $j = 1, \cdots, q, i = 1, \cdots, j$ **do**
5: $\quad v_X(k) = \sqrt{2}\Omega_{ij}$, where Ω_{ij} is the i-th row and j-th column entry of Ω;
6: $\quad k \leftarrow k + 1$;
7: **end for**
8: $d = KX^T v||v||/||vXX^T v||$;
9: **for** $i = 1, \cdots, n - q$ **do**
10: $\quad v_X(k) = d_i$ and $k \leftarrow k + 1$;
11: **end for**
12: return vector $v_X \in \mathbb{R}^{\frac{q(q+1)}{2}+(n-q)}$.

Algorithm 9 Computation of $D2E_X^{\mathcal{F}_v}(N)$

Require: $X \in \mathcal{F}_v$, $v_X \in \mathbb{R}^{\frac{q(q+1)}{2}+(n-q)}$, a positive vector $v \in \mathbb{R}^n$, a function $\beta : \mathbb{R}^{n\times q} \to \mathbb{R}^{n\times q} : A \mapsto [XX_\perp]A$ generated by Algorithm 12;

k=1;
for $j = 1, \cdots, q, i = 1, \cdots, j$ **do**
    $\Omega_{ij} = v_X(k)/\sqrt{2}$, and $\Omega_{ji} = v_X(k)/\sqrt{2}$;
    $k \leftarrow k + 1$;
end for
$\Omega_{ii} = 2\Omega_{ii}$;
for $i = 1, \cdots, n - q$ **do**
    $d_i = v_X(k)$ and $k \leftarrow k + 1$;
end for
$K = dv^T X/||v||$;
return $\beta_X(\begin{bmatrix}\Omega \\ K\end{bmatrix})$.

Algorithm 10 Computation of unit vectors in Householder matrices $(v_1, v_2, \cdots, v_p)$ and sign scalars $(s_1, s_2, \cdots, s_p)$

Require: $Z = [z_1 z_2 \cdots z_p] \in \mathbb{R}^{n\times p}$;

for $i = 1, \cdots, p$ **do**
    Let a denote $-sgn(\tilde{z}_{i1})||\tilde{z}_i||_2$ and define $v_i = (\tilde{z}_i - ae_1)/||\tilde{z}_i - ae_1||_2$ and $s_i = -sgn(\tilde{z}_{i1})$, where $\tilde{z}_i$ is the vector formed by last $n - i + 1$ entries of z_i, $\tilde{z}_{i1}$ is the first entry of $\tilde{z}_i$ and e_1 denotes the first canonical basis of $\mathbb{R}^{n-i+1}$;
    $Z = [z_1 z_2 \cdots z_p] \leftarrow Q_i Z$, where $Q_i = \begin{bmatrix} I_{i-1} & 0 \\ 0 & I_{n-i+1} - 2v - iv_i^T \end{bmatrix}$.
end for
return $(v_1, v_2, \cdots, v_p)$ and $(s_1, s_2, \cdots, s_p)$.

Algorithm 11 Computation of $\alpha_X(A)$

Require: $A \in \mathbb{R}^{n\times p}$, and $V_X = (v_1, v_2, \cdots, v_p)$ and $S_X = (s1, s_2, \cdots, s_p)$ generated by Algorithm 10 with input X;

for $i = 1, \cdots, p$ **do**
    $A \leftarrow Q_i A$, where $Q_i = \begin{bmatrix} I_{i-1} & 0 \\ 0 & I_{n-i+1} - 2v - iv_i^T \end{bmatrix}$
end for
return $diag(s_1, s_2, \cdots, s_p, I_{n-p})A$.

Algorithm 12 Computation of $\beta(A)$

Require: $A \in \mathbb{R}^{n\times p}$, and $V_X = (v_1, v_2, \cdots, v_p)$ and $S_X = (s1, s_2, \cdots, s_p)$ generated by Algorithm 10 with input X;

1: $A \leftarrow diag(s_1, s_2, \cdots, s_p, I_{n-p})A$;
2: **for** $i = p, p-1, \cdots, 1$ **do**
3: $\quad A \leftarrow Q_i A$, where $Q_i = \begin{bmatrix} I_{i-1} & 0 \\ 0 & I_{n-i+1} - 2v - iv_i^T \end{bmatrix}$
4: **end for**
5: return A.

CHAPTER 5

INEXACT ACCELERATED MANIFOLD PROXIMAL GRADIENT OPTIMIZATION METHOD OVER THE FEASIBLE SET $\mathcal{F}_v$

5.1 Inexact Accelerated Manifold Proximal Gradient Optimization Method

The AManPG method in [53] requires the Subproblem (4.13) to be solved exactly for convergence analysis. To speed up the AManPG algorithm over the feasible set $\mathcal{F}_v$, we can apply the semi-smooth Newton method [119], [69] to solve the proximal subproblem approximately. We call the corresponding algorithm as the Inexact AManPG (I-AManPG) algorithm. As analyzed in Chapter 4, the general idea of semi-smooth Newton method is to solve a system of nonlinear equations based on the generalized Jacobian. So, we need to reduce the optimization problem to a system of nonlinear equations in order to use the semi-smooth Newton method. This can be obtained by considering the KKT conditions. Here, we do not require solving the system of nonlinear equations with respect to Λ exactly but only find a Λ such that $||\Psi(\Lambda)||$ is sufficiently small in the sense that

$$||\Psi(\Lambda)|| \leq \frac{\sqrt{4\mu^2 L_g^2 + ||\hat{v}(\Lambda)||^2/2} - 2\mu L_g}{\kappa_u}, \tag{5.1}$$

where $\hat{v}(\Lambda) = P_{T_{x_k}\mathcal{M}}(Prox_{\mu g}(x - \mu(\xi_x - B_x\Lambda)) - x)$ is the resulting search direction η_x. The quantitative accuracy that guarantees global convergence is also given in Section 5.2. The inexact AManPG algorithm is stated in algorithm 13.

A proximal mapping in the Euclidean setting often admits a computationally cheap closed-form solution. However, in the Riemannian setting, the proximal mapping does not usually have a closed form solution due to the existence of an extra linear constraint: $\eta \in \mathrm{T}_{y_k}\mathcal{M}$. The existing Riemannian proximal mappings in [25, 53, 49, 54] are solved by a semi-smooth Newton algorithm. In the theoretical analyses of [25, 53, 49] for global convergence, the Riemannian proximal mappings are assumed to be solved exactly. In [54], an inexact Riemannian proximal gradient (IRPG) is

Algorithm 13 Inexact Accelerated Manifold Proximal Gradient Method (I-AManPG)

Require: Lipschitz constant L_f on ∇f, parameter $\mu > 0$ in the proximal mapping, line search parameter $\sigma \in (0, 1/(8\mu)]$, shrinking parameter in line search $\nu \in (0, 1)$, positive integer N for safeguard, initial iterates Λ_y and Λ_z for the semi-smooth Newton algorithm;

$t_0 = 1$, $y_0 = x_0$, $z_0 = x_0$;
for $k = 0, \ldots$ **do**
  if $\mod(k, N) = 0$ **then** ▷ Invoke safeguard every N iterations
    Invoke Algorithm 14: $[z_{k+N}, x_k, y_k, t_k, \Lambda_z] = Alg7(z_k, x_k, y_k, t_k, F(x_k), \Lambda_z)$;
  end if
  Approximately solve $\eta_{y_k} \approx \operatorname{argmin}_{\eta \in \mathrm{T}_{y_k}\mathcal{M}} \langle \operatorname{grad} f(y_k), \eta\rangle + \frac{1}{2\mu}\|\eta\|^2 + g(y_k + \eta)$ such that (5.10) holds;
  $x_{k+1} = R_{y_k}(\eta_{y_k})$;
  $t_{k+1} = \frac{\sqrt{4t_k^2+1}+1}{2}$;
  Compute

$$y_{k+1} = R_{x_{k+1}}\left(\frac{1-t_k}{t_{k+1}} P_{T_{x_{k+1}}\mathcal{M}}(x_k - x_{k+1})\right);$$

end for

Algorithm 14 Safeguard for Algorithm I-AManPG

Require: $(z_k, x_k, y_k, t_k, F(x_k), \Lambda_z)$; The maximum number of iterations for line search $N_{\max} > 0$;

Ensure: $[z_{k+N}, x_k, y_k, t_k, \Lambda_z]$;

Approximately solve $\eta_{z_k} \approx \operatorname{argmin}_{\eta \in \mathrm{T}_{z_k}\mathcal{M}} \langle \operatorname{grad} f(z_k), \eta\rangle + \frac{1}{2\mu}\|\eta\|^2_{W_{z_k}} + g(z_k + \eta)$ such that (5.10) holds;
Set $\alpha = 1$, and $i_{\text{iter}} = 0$;
while $F(R_{z_k}(\alpha\eta_{z_k})) > F(z_k) - \sigma\alpha\|\eta_{z_k}\|_{\mathrm{F}}^2$ and $i_{\text{iter}} < N_{\max}$ **do**
  $\alpha = \nu\alpha$; $i_{\text{iter}} = i_{\text{iter}} + 1$;
end while
if $i_{\text{iter}} = N_{\max}$ **then**
  Line search fails;
end if
if $F(R_{z_k}(\alpha\eta_{z_k})) < F(x_k)$ **then** ▷ Safeguard takes effect
  $x_k = R_{z_k}(\alpha\eta_{z_k})$, $y_k = R_{z_k}(\alpha\eta_{z_k})$, and $t_k = 1$;
else
  x_k, y_k and t_k keep unchanged;
end if
$z_{k+N} = x_k$; ▷ Update the compared iterate;

Algorithm 15 A regularized semi-smooth Newton Algorithm

Input: $(x, \Lambda, \xi_x), L_g$ is the Lipschitz constant of the function g, the line search parameter $c = 0.001, \tau \in (0,1), \beta > 0, \lambda = 0.1, 0 < \kappa_1 \leq \kappa_2 < 1, 1 < \gamma_1 < \gamma_2, \underline{\lambda} \in (0,1)$;

Output: $[\eta_x, \Lambda]$;

Compute $v(\Lambda) = Prox_{\mu g}(x - (\xi_x - B_x\Lambda)) - x, \hat{v}(\Lambda) = P_{T_xM}(v(\Lambda))$ and $\Psi(\Lambda) = N_x^T(v(\Lambda))$;

while $\kappa_u||\Psi(\Lambda)|| > \sqrt{4\mu^2 L_g^2 + ||\hat{v}(\Lambda)||^2/2} - 2\mu L_g$ **do**

  Compute $\delta = \min(\lambda||\Psi(\Lambda)||, 0.01)$;

  Approximately solve

$$(J_\Psi(\Lambda) + \delta I)[d] = -\Psi(\Lambda)$$

  for d such that $||(J_\Psi(\Lambda) + \delta I)[d] + \Psi(\Lambda)|| \leq \tau \min(1, \lambda||\Psi(\Lambda)||||d||)$;

  $\Lambda_u = \Lambda + d$; ▷ Compute a Newton step

  if $||\Psi(\Lambda_u)|| < \nu||\Psi(\Lambda)||$ **then**

    $\Lambda \leftarrow \Lambda_u$;

  else

    Compute $\rho = -\frac{\mathrm{tr}(\Psi(\Lambda_u)^T d)}{||d||^2}$;

    Update iterate

$$\Lambda \leftarrow \begin{cases} \Lambda_v & if\ \rho \geq \kappa_1\ and\ ||\Psi(\Lambda_v)|| \leq ||\Psi(\Lambda)|| \\ \Lambda_w & if\ \rho \geq \kappa_1\ and\ ||\Psi(\Lambda_v)|| > ||\Psi(\Lambda)|| \\ \Lambda & if\ \rho < \kappa_1, \end{cases}$$

  where

$$\Lambda_v = \Lambda - \frac{\mathrm{tr}(\Psi(\Lambda_u)^T(\Lambda - \Lambda_u))}{||\Psi(\Lambda_u)||^2}\Psi(\Lambda_u), \Lambda_w = \Lambda - \beta\Psi(\Lambda).$$

    Set

$$\lambda \in \begin{cases} (\underline{\lambda}, \lambda) & if\ \rho \neq \kappa_2 \\ [\lambda, \gamma\lambda] & if\ \kappa_1 \leq \rho < \kappa_2 \\ (\gamma_1\lambda, \gamma_2\lambda] & otherwise. \end{cases}$$

  end if

end while

$\eta_x = \hat{v}(\Lambda)$;

proposed, and the theoretical conditions can guarantee local convergence rate is given. Though not solving the Riemannian proximal mapping exactly, IRPG assumes a sufficiently small μ in (4.13) which is usually unknown and estimating μ requires extra effort in practice. In this dissertation, we also solve the Riemannian proximal mapping approximately. The inexact AManPG algorithm in this dissertation avoids the problem of estimating μ in an expensive computation and allows adaptive step size. Also, the motivation is by doing it this way it is more computationally efficient. The need for a convergence proof then follows to maintain the confidence we have for AManPG and IRPG. Specifically, It is shown in Lemma 5.2.2 that if the Riemannian proximal mapping is solved with sufficient accuracy, then the search direction is descent, independently of the choice of μ.

If η_x^* is the exact solution of (4.13) (for conciseness and consistency with Algorithm 15, we omit the subscript k, use x instead of y, and use ξ_x to denote $\operatorname{grad} f(x)$.), then it satisfies

$$\eta_x^* = \operatorname*{argmin}_{\eta} \langle \xi_x, \eta \rangle + \frac{1}{2\mu}\langle \eta, \eta \rangle + g(x+\eta) \quad \text{subject to} \quad \eta \in \mathrm{T}_x \mathcal{M}. \tag{5.2}$$

It follows that $\eta \in \mathrm{T}_x \mathcal{M}$ is equivalent to $B_x^T \eta = 0$. Therefore, the KKT condition for (5.2) is given by

$$\partial_\eta \mathcal{L}(\eta, \Lambda) = 0, \tag{5.3}$$

$$B_x^T \eta = 0, \tag{5.4}$$

where $\mathcal{L}(\eta, \Lambda)$ is the Lagrangian function defined by

$$\mathcal{L}(\eta, \Lambda) = \langle \xi_x, \eta \rangle + \frac{1}{2\mu}\langle \eta, \eta \rangle + g(x+\eta) - \langle \Lambda, B_x^T \eta \rangle. \tag{5.5}$$

Equation (5.3) yields

$$\eta = v(\Lambda) := \operatorname{Prox}_{\mu g}\left(x - \mu(\xi_x - B_x \Lambda)\right) - x, \tag{5.6}$$

where

$$\operatorname{Prox}_{\mu g}(z) = \operatorname*{argmin}_{v \in \mathbb{R}^{n\times p}} \frac{1}{2}\|v - z\|^2 + \mu g(v) \tag{5.7}$$

denotes the proximal mapping. Substituting (5.6) into (5.4) yields that

$$\Psi(\Lambda) := B_x^T \left(\operatorname{Prox}_{\mu g}\left(x - \mu(\xi_x - B_x \Lambda)\right) - x\right) = 0, \tag{5.8}$$

which is a system of nonlinear equations with respect to Λ. Therefore, to solve (5.2), one can first find the root of (5.8) and substitute it back to (5.6) to obtain η_x^*.

The equation (5.8) can be solved efficiently by a regularized semi-smooth Newton method in Algorithm 15 [119]. Analogous to the classical Newton method, the estimation of Λ is updated by $\Lambda_{k+1} = \Lambda_k + d_k$, where d_k is computed by solving a Newton equation, i.e.,

$$J_\Psi(\Lambda_k)[d] = -\Psi(\Lambda_k), \tag{5.9}$$

where $J_\Psi(\Lambda_k)$ is a generalized Jacobian of Ψ. Moreover, by the chain rule, we have

$$J_\Psi(\Lambda_k)[d] = B_x^T \left(\partial \operatorname{Prox}_{\mu g}\left(x - \mu(\xi_x - B_x\Lambda_k)\right) \odot (\mu B_x d)\right),$$

where $\partial \operatorname{Prox}_{\mu g}(\cdot)$ denotes the generalized Clarke subdifferential of $\operatorname{Prox}_{\mu g}(\cdot)$ and $\odot$ denotes the entrywise product of two matrices. Note that when $g(x) = \lambda\|x\|_1$, the generalized Clarke subdifferential of $\operatorname{Prox}_{\mu g}(\cdot)$ can be computed in an entrywise manner [26, 118, 68]. Here we do not require solving (5.8) exactly but only find a Λ such that $||\Psi(\Lambda)||$ is sufficiently small in the sense that

$$||\Psi(\Lambda)|| \leq \frac{\sqrt{4\mu^2 L_g^2 + ||\hat{v}(\Lambda)||^2/2} - 2\mu L_g}{\kappa_u}, \tag{5.10}$$

where $\hat{v}(\Lambda) = P_{T_{x_k}\mathcal{M}}(Prox_{\mu g}(x - \mu(\xi_x - B_x\Lambda)) - x)$ is the resulting search direction η_x.

5.2 Global Convergence Analysis of I-AManPG

Lemma 5.2.1 states the key result used to prove the global convergence. In [25, Lemma 5.1], the inequality (5.12) is proven up to a coefficient under the assumption that the subproblem (4.13) is solved exactly. Here, it is shown that if the subproblem is solved accurately enough such that (5.11) holds, then we also have the inequality (5.12).

Lemma 5.2.1. *Let $\ell_x(\eta) = \langle \xi_x, \eta\rangle + \frac{1}{2\mu}\langle \eta, \eta\rangle + g(x+\eta)$. Let ϵ denote $\Psi(\Lambda) = B_x^T v(\Lambda)$. We then have*

$$g(x) \geq \langle \xi_x, \hat{v}(\Lambda)\rangle + \frac{1}{2\mu}\|\hat{v}(\Lambda)\|^2 + g(x + \hat{v}(\Lambda)) - (2L_g + \frac{1}{2\mu}\|\epsilon\|)\|\epsilon\|,$$

where $\hat{v}$ is defined in (5.10)*. Furthermore, if ϵ is sufficiently close to 0 in the sense that*

$$\|\epsilon\| \leq \sqrt{4\mu^2 L_g^2 + \|\hat{v}(\Lambda)\|^2/2} - 2\mu L_g, \tag{5.11}$$

then it holds that

$$\ell_x(\alpha\hat{v}(\Lambda)) - \ell_x(0) \leq -\left(\frac{\alpha(1-2\alpha)}{4\mu}\right)\|\hat{v}(\Lambda)\|^2, \quad \forall \alpha \in [0,1]. \tag{5.12}$$

Proof. Consider the optimization problem

$$\min_{B_x^T\eta=\epsilon} \ell_x(\eta). \tag{5.13}$$

Its KKT condition is given by

$$\partial_\eta \mathcal{L}(\eta,\Lambda) = 0, \qquad B_x^T\eta = \epsilon,$$

which is satisfied by $v(\Lambda)$ defined in (5.6). Therefore, $v(\Lambda)$ is the minimizer of $\ell_x(\eta)$ over the set $\mathcal{S} = \{v : B_x^T v = \epsilon\}$, i.e.,

$$v(\Lambda) = \operatorname*{argmin}_{v\in\mathcal{S}} \ell_x(\eta) = \langle \xi_x, \eta\rangle + \frac{1}{2\mu}\langle \eta,\eta\rangle + g(x+\eta). \tag{5.14}$$

Define the vector $v_0 = B_x\epsilon$. It can be easily verified that $B_x^T v_0 = B_x^T B_x \epsilon = \epsilon$. Therefore, it holds that $v_0 \in \mathcal{S}$. By $\frac{1}{\mu}$-strong convexity of ℓ_x, we have

$$\ell_x(v_0) \geq \ell_x(v(\Lambda)) + \langle \partial\ell_x(v(\Lambda)), v_0 - v(\Lambda)\rangle + \frac{1}{2\mu}\|v_0 - v(\Lambda)\|^2. \tag{5.15}$$

From the optimality condition of Problem (5.13), we have that $0 \in P_{\mathrm{T}_\eta\mathcal{S}}\partial\ell_x(v(\Lambda))$. Since $\mathrm{T}_\eta\mathcal{S} = \{u : B_x^T u = 0\}$ and $B_x^T(v_0 - v(\Lambda)) = \epsilon - \epsilon = 0$, it holds that $v_0 - v(\Lambda) \in \mathrm{T}_\eta\mathcal{S}$. Therefore, we have

$$0 \in \langle \partial\ell_x(v(\Lambda)), v_0 - v(\Lambda)\rangle. \tag{5.16}$$

It follows from (5.15) and (5.16) that

$$\ell_x(B_x\epsilon) \geq \ell_x(v(\Lambda)) + \frac{1}{2\mu}\|v(\Lambda) - B_x\epsilon\|^2. \tag{5.17}$$

Substituting the definition of ℓ_x into inequality (5.17) and noting $\epsilon = B_x^T v(\Lambda)$, we have that

$$\begin{aligned}\frac{1}{2\mu}\|B_xB_x^Tv(\Lambda)\|^2 + g(x + B_xB_x^Tv(\Lambda)) \geq \\ \langle \xi_x, v(\Lambda)\rangle + \frac{1}{2\mu}\|v(\Lambda)\|^2 + g(x+v(\Lambda)) + \frac{1}{2\mu}\|v(\Lambda) - B_xB_x^Tv(\Lambda)\|.\end{aligned} \tag{5.18}$$

It follows that

$$
\begin{aligned}
g(x) \geq & \langle \xi_x, P_{\mathrm{T}_x \mathcal{M}} v(\Lambda) \rangle + \frac{1}{2\mu} \|P_{T_x \mathcal{M}} v(\Lambda)\|^2 + g(x + v(\Lambda)) + g(x) \\
& - g(x + B_x \epsilon) - \frac{1}{2\mu} \|B_x B_x^T v(\Lambda)\|^2 \\
\geq & \langle \xi_x, \hat{v}(\Lambda) \rangle + \frac{1}{2\mu} \|\hat{v}(\Lambda)\|^2 + g(x + \hat{v}(\Lambda)) + g(x + v(\Lambda)) \\
& - g(x + (I - B_x B_x^T) v(\Lambda)) + g(x) - g(x + B_x \epsilon) - \frac{1}{2\mu} \|B_x \epsilon\|^2 \\
\geq & \langle \xi_x, \hat{v}(\Lambda) \rangle + \frac{1}{2\mu} \|\hat{v}(\Lambda)\|^2 + g(x + \hat{v}(\Lambda)) - |g(x + v(\Lambda)) - g(x + v(\Lambda) - B_x \epsilon)| \\
& - |g(x) - g(x + B_x \epsilon)| - \frac{1}{2\mu} \|B_x \epsilon\|^2 \\
\geq & \langle \xi_x, \hat{v}(\Lambda) \rangle + \frac{1}{2\mu} \|\hat{v}(\Lambda)\|^2 + g(x + \hat{v}(\Lambda)) - (2L_g + \frac{1}{2\mu} \|B_x \epsilon\|) \|B_x \epsilon\|, \qquad (5.19)
\end{aligned}
$$

where the first inequality follows from (5.18) and $\|v(\Lambda)\|^2 \geq 0$, the second inequality follows from $\hat{v}(\Lambda) = P_{\mathrm{T}_x \mathcal{M}} v(\Lambda) = (I - B_x B_x^T) v(\Lambda)$, and the fourth inequality follows from the Lipschitz continuity of g with Lipschitz constant L_g. This completes the proof for the first result.

Since g is convex, we have

$$
g(x + \alpha \hat{v}(\Lambda)) - g(x) = g(\alpha(x + \hat{v}(\Lambda)) + (1 - \alpha)x) - g(x) \leq \alpha(g(x + \hat{v}(\Lambda)) - g(x)). \qquad (5.20)
$$

Combining (5.19) and (5.20) yields

$$
\begin{aligned}
\ell_x(\alpha \hat{v}(\Lambda)) - \ell_x(0) = & \langle \xi_x, \alpha \hat{v}(\Lambda) \rangle + \frac{1}{2\mu} \|\alpha \hat{v}(\Lambda)\|^2 + g(x + \alpha \hat{v}(\Lambda)) - g(x) \\
\leq & \alpha \left(\langle \xi_x, \hat{v}(\Lambda) \rangle + \frac{\alpha}{2\mu} \|\hat{v}(\Lambda)\|^2 + g(x + \hat{v}(\Lambda)) - g(x) \right) \\
\leq & \alpha \left(\frac{\alpha}{2\mu} \|\hat{v}(\Lambda)\|^2 - \frac{1}{2\mu} \|\hat{v}(\Lambda)\|^2 + (2L_g + \frac{1}{2\mu} \|B_x \epsilon\|) \|B_x \epsilon\| \right). \qquad (5.21)
\end{aligned}
$$

By $\|\epsilon\| = \|B_x \epsilon\| \leq \sqrt{4\mu^2 L_g^2 + \|\hat{v}(\Lambda)\|^2/2} - 2\mu L_g$, we have

$$
(2L_g + \frac{1}{2\mu} \|B_x \epsilon\|) \|B_x \epsilon\| \leq \frac{1}{4\mu} \|\hat{v}(\Lambda)\|^2. \qquad (5.22)
$$

The second result follows from (5.21) and (5.22). Finally, (5.11) follows from the definition of $\Psi(\Lambda)$. □

Lemma 5.2.2 implies that the while loop in Step 3 of Algorithm 7 terminates in a finite number of iterations. Given (5.12) in Lemma 5.2.1, the proof of Lemma 5.2.2 follows the same steps as that of [25, Lemma 5.2] is therefore omitted.

Lemma 5.2.2. *Suppose Assumption 4.2.1 holds. Then for any $\mu > 0$, there exists a constant $\bar{\alpha} \in (0, 1]$ such that for any $0 < \alpha < \bar{\alpha}$, Step 3 of Algorithm 7 is satisfied, and the sequence $\{z_k\}$ generated by Algorithm 6 satisfies*

$$F(R_{z_k}(\alpha\eta_{z_k})) - F(z_k) \leq -\frac{\alpha}{8\mu}\|\eta_{z_k}\|^2.$$

Moreover, the step size $\alpha > \rho\bar{\alpha}$ for all k.

Though the subproblem is solved inexactly, a zero search direction given by $\hat{v}(\Lambda)$ with condition (5.10) implies that the current iterate x is a stationary point, which coincides with [25, Lemma 5.3].

Lemma 5.2.3. *If $\eta_x = \hat{v}(\Lambda) = 0$, then x is a stationary point of Problem 4.11.*

Proof. If $\eta_x = \hat{v}(\Lambda) = 0$, then by (5.10), we have that $\Psi(\Lambda) = 0$, which implies that the subproblem (5.2) is solved exactly and $\eta_x^* = 0$. By [25, Lemma 5.3], x is a stationary point of Problem 4.11. □

The main convergence result is given in Theorem 5.2.4. The proof follows the spirit of [53, Theorem 1]. Here, we only highlight their differences.

Theorem 5.2.4. *Suppose Assumption 4.2.1 holds, then any accumulation point of the sequence $\{z_0, z_N, z_{2N}, \ldots, z_{iN}, \ldots\}$ generated by Algorithm 13 is a stationary point, i.e., if z_* is an accumulation point of the above sequence, then $0 \in P_{\mathrm{T}_{z_*}\mathcal{M}}\partial F(z_*)$.*

Proof. Since the subscript of z_k in Algorithm 13 is a multiple of N, we use $\{\tilde{z}_i\}$ to denote $\{z_k\}$, where $\tilde{z}_i = z_{iN}$. Let $(\eta_{\tilde{z}_i}, \Lambda_{\tilde{z}_i})$ denote the output of Algorithm 15 when the input is $(\tilde{z}_i, \Lambda_{\tilde{z}_{i-1}}, \operatorname{grad} f(\tilde{z}_i))$, i.e., the input and output of Step 1 of Algorithm **??**.

By the safeguard in Algorithm **??** and Lemma 5.2.2, we have

$$F(\tilde{z}_{i+1}) - F(\tilde{z}_i) \leq -\frac{\rho\bar{\alpha}}{8\mu}\|\eta_{\tilde{z}_k}\|^2.$$

Since F is continuous and $\mathcal{F}_v$ is compact, the function F is bounded from below. It follows that

$$\infty > \sum_{i=0}^{\infty} F(\tilde{z}_i) - F(\tilde{z}_{i+1}) \geq \frac{\rho\bar{\alpha}}{8\mu}\|\eta_{\tilde{z}_k}\|^2, \tag{5.23}$$

which implies

$$\lim_{i\to\infty} \|\eta_{\tilde{z}_k}\| = 0. \tag{5.24}$$

By (5.24), (5.10), and $\eta_{\tilde{z}_k} = \hat{v}(\tilde{z}_i)$, we have

$$\lim_{i\to\infty} \|\Psi(\Lambda_{\tilde{z}_i})\| = 0. \tag{5.25}$$

Since $\hat{v}(\Lambda_{\tilde{z}_i}) = P_{\mathrm{T}_{\tilde{z}_i}\mathcal{M}} v(\Lambda_{\tilde{z}_i}) = v(\Lambda_{\tilde{z}_i}) - B_{\tilde{z}_i}\Psi(\Lambda_{\tilde{z}_i})$, we have

$$\|v(\Lambda_{\tilde{z}_i})\| = \|\hat{v}(\Lambda_{\tilde{z}_i}) + B_{\tilde{z}_i}\Psi(\Lambda_{\tilde{z}_i})\| \le \|\hat{v}(\Lambda_{\tilde{z}_i})\| + \|B_{\tilde{z}_i}\Psi(\Lambda_{\tilde{z}_i})\| = \|\eta_{\tilde{z}_k}\| + \|\Psi(\Lambda_{\tilde{z}_i})\|. \tag{5.26}$$

Combining (5.24), (5.25) and (5.26) yields

$$\lim_{i\to\infty} \|v(\Lambda_{\tilde{z}_i})\| = 0. \tag{5.27}$$

By (5.14), we have

$$v(\Lambda_{\tilde{z}_i}) = \operatorname*{argmin}_{\eta\in\mathcal{S}_{\tilde{z}_i}} \ell_x(\eta) = \langle \operatorname{grad} f(\tilde{z}_i), \eta\rangle + \frac{1}{2\mu}\langle \eta, \eta\rangle + g(\tilde{z}_i + \eta), \tag{5.28}$$

where $\mathcal{S}_{\tilde{z}_i} = \{v : B_{\tilde{z}_i}^T v = B_{\tilde{z}_i}^T v(\Lambda_{\tilde{z}_i})\}$. Using (5.27) and (5.28) and following the steps in the proof of [53, Theorem 1], we have that any accumulation point of $\{\tilde{z}_i\}$ is a stationary point. □

CHAPTER 6

INEXACT ACCELERATED MANIFOLD PROXIMAL GRADIENT OPTIMIZATION FOR APPLICATIONS

In this chapter, we design some numerical experiments solving the community detection, k-means model, discriminative k-means model and normalized cut problems to test our algorithms. We denote Algorithm 13 by I-AManPG and Algorithm 13 by E-AManPG with the condition that $\|\Psi(\Lambda)\| \leq 10^{-10}$. Therefore, we can view I-AManPG as an inexact Riemannian proximal gradient method while E-AManPG is essentially an exact one since E-AManPG solves the Riemannian proximal mapping to high accuracy.

The parameters L_f and λ are problem-dependent and are specified later. The other parameters are set to be $\mu = 1/L_f$, $\sigma = 10^{-4}$, $\nu = 0.5$, $N = 5$, $\Lambda_y = \Lambda_z = 0$, $N_{\max} = 5$, $\nu_{\text{SSN}} = 0.9999$, $\beta = 0.1$, $\kappa_1 = 0.2$, $\kappa_2 = 0.75$, $\gamma_1 = 2$, $\gamma_2 = 5$, $\underline{\lambda} = 10^{-5}$, and the initial value $\epsilon_\Psi = 1$.

Unless otherwise indicated, I-AManPG and E-AManPG stop if the value of $\|\eta_{z_k}\|$ reduces at least by a factor of 10^3. The last iterate is projected to the set $\mathcal{A}_v^{n,q}$ by the projection

$$P_{\mathcal{A}_v^{n,q}}(X) = \text{diag}(v) P_{\mathcal{A}_{\mathbf{1}_n}^{n,q}}(\text{diag}(v)^{-1} X)$$

where $P_{\mathcal{A}_{\mathbf{1}_n}^{n,q}}(X) = \left[\frac{b_1}{\|b_1\|} \quad \frac{b_2}{\|b_2\|} \quad \cdots \quad \frac{b_q}{\|b_q\|}\right]$, $b_j \in \mathbb{R}^n$ for $j = 1, 2, \ldots, q$, and

$$(b_j)_i = \begin{cases} 1 & \text{if } X_{ij} \text{ has the largest magnitude in the } i\text{-th row;}^1 \\ 0 & \text{otherwise.} \end{cases}$$

If the i-th row j-th column of $P_{\mathcal{A}_v^{n,q}}(X)$ is not zero, then this implies that the i-th object is in the j-th cluster.

I-AManPG and E-AManPG are implemented in MATLAB. All the experiments are performed on a MAC platform with 1.4 GHz Quad-Core Intel Core i5.

[1] If there is a tie in the i-th row, then j can be any one with the largest magtinude.

6.1 Inexact Accelerated Manifold Proximal Gradient Optimization for Community Detection

In this section, we evaluate the performance of community detection by optimizing the formulation

$$\min_{X \in \mathcal{F}_{\mathbf{1}_n}} -\text{trace}(X^T M X) + \lambda \|X\|_1, \tag{6.1}$$

with I-AManPG algorithm. Firstly, we compare the efficiency of E-AManPG and I-AManPG on the community detection problem. Then we evaluate the empirical performance of I-AManPG on community detection by comparing it with some classical algorithms, such as Danon et al.'s algorithm [28], the Louvain method [12] and Newman's spectral optimization method [82].

6.1.1 Data Sets

We solve the community detection problems on synthetic LFR benchmark networks which are introduced in [65]. For more details, see Section 3.3.1.

The second collection of benchmark data sets consist of four real-world networks. The first three real-world networks are the same as in Section 3.3.2, that is, American college football network, Zachary's karate club network and the Polbook network. The fourth network was generated using email data from a large European research institution [122] [67]. It is generated from anonymized information about all incoming and outgoing email between members of the research institution. There is an edge (u, v) in the network if person u sent person v at least one email. The e-mails only represent communication between institution members (the core), and the dataset does not contain incoming messages from or outgoing messages to the rest of the world. The dataset also contains "ground-truth" community memberships of the nodes. Each individual belongs to exactly one of 42 departments at the research institute. This network represents the "core" of the email-EuAll network, which also contains links between members of the institution and people outside of the institution (although the node IDs are not the same).

6.1.2 Compare the Efficiency of I-AManPG and E-AManPG on LFR benchmarks networks

In this section, we compare the efficiency of I-AManPG and E-AManPG method on detecting community structures of LFR benchmark networks [65]. Throughout this section, the parameters

τ_1, τ_2, and μ_{LFR} are set to -2, -1, and 0.1 respectively. Four sets of other parameters are given as follows

- $N = 500$, $d_{ave} = 10$, $d_{max} = 20$, $N_c = 50$, $n_c = 10$;
- $N = 1000$, $d_{ave} = 20$, $d_{max} = 40$, $N_c = 100$, $n_c = 10$;
- $N = 5000$, $d_{ave} = 40$, $d_{max} = 80$, $N_c = 500$, $n_c = 10$;
- $N = 10000$, $d_{ave} = 40$, $d_{max} = 80$, $N_c = 1000$, $n_c = 10$;

where N denotes the number of nodes, d_{ave} denotes the average node degree, d_{max} denotes the maximum node degree, N_c denotes the number of nodes that all communities have, and n_c denotes the number of communities. The balancing parameter λ in (6.1) is set to 0.3.

From the cost function $-\text{trace}(X^T MX) + \lambda||X||_1$, and $\lambda > 0$ is a tuning parameter, we can deduce $\lambda = \text{trace}(X^T MX)/||X||_1 \simeq \max eig(M) * p/n$. To explain the directly chosen value of λ 0.3, we can compute the $\max eig(M) * p/n$ on LFR benchmarks as in Table 6.1. The average of all the values of $\max eig(M) * p/n$ is 0.324.

Table 6.1: $\max eig(M) * p/n$ for ground-truth partition for LFR benchmark n=1000, p=20.

	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
$\max eig(M) * p/n$	0.4969	0.4493	0.4023	0.3565	0.3119	0.2681	0.2321	0.2042	0.1947

In Table 6.2, an average result of 10 random runs for the graphs generated randomly as LFR benchmark networks. I-A denotes I-AManPG and E-A denotes E-AManPG. n is the number of nodes N, q is the number of communities n_c; The notation iter, SSNiter, nf, ng, nR, nSG, F, $\frac{\|\eta_{z_k}\|}{\|\eta_{z_0}\|}$, and time, respectively, denote the number of iterations in AManPG, the number of iterations in semi-smooth Newton method, the number of function evaluations, the number of gradient evaluations, the number of retraction evaluations, the number of safeguards (Step 9) that are taken, the function value at the final iterate, the reduction of the norm of search directions $\frac{\|\eta_{z_k}\|}{\|\eta_{z_0}\|}$, and the computational time in seconds. The subscript k denotes a scale of 10^k. As shown in Table 6.2, I-AManPG and E-AManPG find the same solutions in the sense that their function values are the same up to three significant digits. Actually, though not reported in the table, we find that, in our experimental setting, both I-AManPG and E-AManPG always converge to the same

Table 6.2: Compare the efficiency of I-AManPG and E-AManPG on LFR benchmark networks.

	I-A	E-A	I-A	E-A	I-A	E-A	I-A	E-A
(n, q)	$(500, 10)$		$(1000, 10)$		$(5000, 10)$		$(10000, 10)$	
iter	64	52	50	59	63	58	55	55
SSNiter	28	212	13	248	34	311	52	330
nf	143	115	112	131	140	128	123	122
ng	83	65	64	73	81	72	71	68
nR	142	114	111	130	139	127	122	121
nSG	4	14	2	15	4	13	3	10
F	-67.00	-67.0	-149	-149	-284	-284	-251	-251
$\frac{\|\eta_{z_k}\|}{\|\eta_{z_0}\|}$	7.0_{-4}	5.7_{-4}	5.5_{-4}	5.1_{-4}	6.3_{-4}	5.8_{-4}	5.2_{-4}	6.9_{-4}
time	0.15	0.31	0.17	0.75	0.84	3.03	1.54	5.19
NMI	1	1	1	1	1	1	1	1
AMI	1	1	1	1	1	1	1	1
Modularity	0.7998	0.7998	0.7998	0.7998	0.8001	0.8001	0.8000	0.8000
Purity	1	1	1	1	1	1	1	1

solution which always represents the ground truth partition. I-AManPG and E-AManPG find the same partitioning in terms of the NMI, AMI, modularity, and purity. In addition, I-AManPG takes less work on solving the Riemannian proximal mapping in the sense that the number of semi-smooth Newton iterations in each outer iteration is small compared to E-AManPG. Moreover, less accuracy for solving the Riemannian proximal mapping does not influence the number of outer iterations significantly. Therefore, I-AManPG is more efficient than E-AManPG in terms of computational time. In later comparisons, we use I-AManPG as the representative method.

6.1.3 Compare the Effectiveness of the Model (6.1) and Existing Community Detection Methods

In this section, community detection by the optimization model (6.1) with I-AManPG is compared to three state-of-the-art methods: Danon et al.'s algorithm [28], the Louvain method [12] and Newman's spectral optimization method [82].

These algorithms all aim to maximize the modularity $Q = \frac{1}{2m}\operatorname{tr}(X^T M X)$ over the set of indicator matrices, where m is the number of edges, and an indicator matrix X is defined by $X \geq 0, X^T X$ is diagonal, and each row of X has an entry being one. Each indicator matrix specifies a partitioning of the nodes into communities. Danon et al.'s algorithm is a variant algorithm of Newman's fast greedy method [80] which assumes that edges are not initially present and are

added one by one by choosing the edge such that this partition gives the maximum increase of modularity with respect to the previous configuration. Louvain method is divided into two phases that are repeated iteratively. The first phase creates intermedia-communities by merging nodes such that the modularity increases. In the second phase, a smaller graph, called reduced graph, is created where each node in this graph represents an intermedia-community. Newman's spectral optimization method computes the eigenvectors of the modularity matrix M corresponding to the largest positive eigenvalues. The nodes are grouped into two parts based on the signs of the component of the eigenvectors. The process is then repeated for each of the parts until making a zero or negative contribution to the total modularity.

To make fair comparisons, we use public-available Matlab implementations of these algorithms. The codes for Danon et al.'s algorithm, Louvain's algorithm, and Newman's spectral method are respectively from [61], [100] and [14]. Note that the codes for Newman's spectral method in [14] do not embed the fine-tuning stage and use a different stopping criteria as in paper [82]. We modified the stopping criteria in the codes such that it has the same stopping criteria as in paper [82]. The codes in [14] use dense matrix computations, and we modified them with significantly more efficient sparse computations. In this way, the computational efficiency was improved to the point where it produced times that were reasonable to include in these comparisons.

To compare the effectiveness of the four methods, we use three quality measurements, which are the normalized mutual information (NMI) [29], the adjusted mutual information (AMI) [114], and the purity [74]. The descriptions of NMI and AMI are in Section 3.3.3.

Given two partitions X and Y of N nodes, the purity is given by

$$\text{purity}(X, Y) = \frac{1}{N} \sum_k \max_j |X_k \cap Y_j|, \tag{6.2}$$

where X_k denotes the set of nodes in k-th community of partition X, and likewise for Y_j, and $|X_k \cap Y_j|$ denotes the number of nodes in $X_k \cap Y_j$. The value of purity is also between 0 and 1. From the definition of purity 6.2, we can see that

- Purity is not symmetric;
- If the graphs are nested, then the purity is 1 if we compare the cluster assignment with larger number of clusters to the clusters with smaller number of cluster assignment.

Algorithm 16 Algorithm for purity modification

```
if max(cluster1)≥ max(cluster2) then
    purity=purity(cluster1, cluster2)
    else
    purity=purity(cluster2,cluster1)
end if
```

So, we modify purity by the following Algorithm 16.

Even though we cannot use purity to trade off the quality of the clustering against the number of clusters, we can use it to measure whether two partitions are nested or not. The closer it is to one, the better the two partitions are nested. In our numerical experiments, the ground truth is known and therefore the computed partition is compared to the ground truth. When we do not have the same number of clusters as the ground-truth assignment to clusters, purity checks if the assignment at least preserves the ground-truth in a hierarchical fashion and therefore can potentially be recovered with further aggregation or division. The latter of which we use in the recursive algorithm in Chapter 7.

LFR Benchmarks. For LFR benchmark networks that we use in this section, the parameters τ_1, τ_2, N, d_{ave}, d_{max}, N_c, and n_c are respectively set to -2, -1, 1000, 20, 40, 50, and 20. The value of λ in (6.1) is 0.3. The numerical results with multiple values of μ_{LFR} are reported in Table 6.3.

Each result in Table 6.3 is an average result of 10 random runs for the graphs generated randomly as LFR benchmark networks. From the results in Table 6.3, we observe that when $\mu_{\text{LFR}} = 0$, I-AManPG yields NMI = AMI = purity = 1, the same modularity value and the same assignment to $q_{true} = 20$ strongly connected communities. The Louvain method also has the same results with ground-truth communities while Danon et al.'s algorithm and Newman's spectral algorithm can get the results which are very close to the ground-truth communities, specifically they can detect exactly ground-truth communities for 9 of 10 random LFR graphs. When μ_{LFR} takes 0.1 to 0.4, I-AManPG and Louvain algorithm can detect the exact ground-truth communities. When $\mu_{\text{LFR}} = 0.5, 0.6$, I-AManPG can get results very close to ground-truth partitions and the results are competitive results with the Louvain algorithm, but with less time. When $\mu_{\text{LFR}} = 0.7, 0.8$, the results for all of these four algorithms are far away from the ground-truth partitions because the community structures in these cases are not strong. Danon's algorithm and Newman spectral algorithm detect

Table 6.3: Compare the effectiveness of I-AManPG to other state-of-the-art methods. q_c is the computed number of communities, q is the input parameter for I-AManPG and "force_q" algorithms denote the algorithms by adding a forcing criteria such that the algorithms stop when q_c is smaller than or equal to $q_{true} = 20$.

		μ_{LFR}								
		0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
Danon	NMI	0.9998	0.9891	0.9394	0.8504	0.7331	0.5808	0.3781	0.1412	0.0548
	AMI	0.9998	0.9870	0.9166	0.7922	0.6399	0.4736	0.2878	0.0935	0.0215
	Mod.	0.9496	0.8436	0.7225	0.5920	0.4687	0.3452	0.2458	0.1892	0.1814
	purity	0.9999	0.9938	0.9739	0.9414	0.9058	0.8344	0.6886	0.4269	0.3095
	time	2.8304	2.8597	2.8199	2.7648	2.7287	2.8318	2.7739	2.7625	2.7318
	q_c	20	20	18	15	11	9	7	7	8
Danon_force_q	NMI	0.9998	0.9889	0.9397	0.8494	0.7309	0.5794	0.3939	0.1778	0.0954
	AMI	0.9998	0.9870	0.9201	0.7962	0.6415	0.4727	0.2950	0.1035	0.0282
	Mod.	0.9496	0.8433	0.7209	0.5868	0.4615	0.3394	0.2431	0.1872	0.1794
	purity	0.9999	0.9935	0.9714	0.9349	0.8954	0.8198	0.6778	0.4201	0.3065
	time	2.8353	2.9606	2.8222	2.8068	2.8055	2.8959	2.8183	2.8459	2.7432
	q_c	20	20	20	20	20	20	20	20	20
Louvain	NMI	1.0000	1.0000	1.0000	1.0000	1.0000	0.9987	0.9805	0.2862	0.0784
	AMI	1.0000	1.0000	1.0000	1.0000	1.0000	0.9974	0.9652	0.2249	0.0358
	Mod.	0.9497	0.8499	0.7503	0.6500	0.5499	0.4496	0.3477	0.2098	0.1967
	purity	1.0000	1.0000	1.0000	1.0000	1.0000	0.9999	0.9950	0.4734	0.2660
	time	0.5444	0.7291	1.3703	1.8963	2.6797	3.3418	4.5768	9.2669	8.8130
	q_c	20	20	20	20	20	20	19	11	11
Louvain_force_q	NMI	1.0000	1.0000	1.0000	1.0000	1.0000	0.9987	0.9805	0.2981	0.0847
	AMI	1.0000	1.0000	1.0000	1.0000	1.0000	0.9974	0.9652	0.2382	0.0392
	Mod.	0.9497	0.8499	0.7503	0.6500	0.5499	0.4496	0.3477	0.2098	0.1967
	purity	1.0000	1.0000	1.0000	1.0000	1.0000	0.9999	0.9950	0.4642	0.2418
	time	0.5440	0.7478	1.0333	1.2042	1.7002	2.0761	2.7676	5.4529	5.5061
	q_c	20	20	20	20	20	20	19	12	12
Newman_Eig Sparse	NMI	0.9988	0.7225	0.7132	0.6760	0.5498	0.3912	0.2807	0.1340	0.0497
	AMI	0.9985	0.6521	0.6396	0.6122	0.4704	0.3071	0.2098	0.0907	0.0235
	Mod.	0.9482	0.5493	0.5057	0.4157	0.3051	0.2379	0.1917	0.1578	0.1461
	purity	0.9994	0.7828	0.8134	0.7460	0.6723	0.6366	0.5654	0.4367	0.3594
	time	0.6346	0.4391	0.4197	0.4522	0.4140	0.3333	0.3565	0.3130	0.3026
	q_c	20	24	21	19	15	9	7	6	6
Newman_Eig_force_q Sparse	NMI	0.9988	0.6831	0.6787	0.6674	0.5498	0.3912	0.2807	0.1340	0.0497
	AMI	0.9985	0.5996	0.5991	0.6026	0.4704	0.3071	0.2098	0.0907	0.0235
	Mod.	0.9482	0.4747	0.4463	0.4004	0.3051	0.2379	0.1917	0.1578	0.1461
	purity	0.9994	0.7998	0.8001	0.7358	0.6723	0.6366	0.5654	0.4367	0.3594
	time	0.6458	0.4666	0.4373	0.4528	0.4231	0.3418	0.3656	0.3213	0.3114
	q_c	20	18	17	18	15	9	7	6	6
I-AManPG	NMI	1.0000	1.0000	1.0000	1.0000	1.0000	0.9998	0.9600	0.4517	0.1294
	AMI	1.0000	1.0000	1.0000	1.0000	1.0000	0.9998	0.9539	0.4037	0.0563
	Mod.	0.9497	0.8499	0.7503	0.6500	0.5499	0.4498	0.3416	0.1735	0.1113
	purity	1.0000	1.0000	1.0000	1.0000	1.0000	0.9999	0.9679	0.5605	0.3044
	time	0.6357	0.4693	0.5870	0.9494	0.6749	0.4720	1.0332	1.6307	1.6757
	q	20	20	20	20	20	20	20	20	20

relatively inaccurate communities and relatively small qualifying external or internal measurements, i.e., NMI, AMI, purity and modularity for all noisy cases. From the computational times, we can see that I-AManPG requires relatively less time than the others. The reason that the computational time of the Newman's spectral method is less than I-AManPG is that Newman's spectral method is one recursive algorithm. When we compare it to the recursive I-AManPG algorithm in Chapter 7, the recursive I-AManPG takes less time than Newman's spectral method. In this sense, it motivates the generation of the recursive I-AManPG. It is worth noting that the number of edges m does not change much and only the distribution of edges changes a lot as the mixing parameter increases. So, the computational time for Danon's algorithm and Newman's spectral method do not change much as the mixing parameter increases because the computational time of these two algorithms depends more on m rather than on the distribution of edges.

As the mixing parameter increases, the difficulty level of detecting the number of communities q increases as well. I-AManPG requires the desired number of communities as a parameter value and the choice of an initial q and the development of a dynamic adaptation strategy are key ongoing tasks for I-AManPG. To make more fair comparisons with other algorithms, we modified the stopping criteria of the algorithms to which I-AManPG is compared such that they try to reach the number of correct communities of the ground-truth partition which is added to their parameter list. Note that the computed number of communities is not necessarily the same with q_{true} even though the modification to the stopping criteria attempts to force this. The computed numbers still depends on the properties of the specified algorithm and the particular sequence of cluster assignments it produces for the given graph and parameter values. Even when we force the number of communities to be 20 for the other three methods, we observe that the partitions from the other three methods are not as close as the ground-truth partitions as the partitions from I-AManPG by comparing the results of NMI, AMI and purity.

To make more fair comparisons with the other algorithms, we tried different choices of number of communities q as inputs for the I-AManPG algorithm. The implementation on the modified versions of other algorithms was run on 10 random generated LFR benchmark networks for different mixing parameters μ and the results are shown in Table 6.4.

Consider the results of different algorithms for the cases with the same computed number of communities. From the results in Table 6.3 and Table 6.4, we observe that NMI, AMI, modularity

Table 6.4: Test the effectiveness of I-AManPG for different input parameter q.

		μ_{LFR}								
		0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
	NMI	0.7178	0.7178	0.7178	0.7176	0.7137	0.6966	0.6230	0.2957	0.0795
	AMI	0.5452	0.5452	0.5452	0.5451	0.5428	0.5327	0.4856	0.2202	0.0324
I-AManPG	Mod.	0.6924	0.6198	0.5474	0.4733	0.4005	0.3251	0.2531	0.1534	0.1007
	purity	1.0000	1.0000	1.0000	0.9999	0.9971	0.9842	0.9243	0.6363	0.5118
	time	0.1939	0.2175	0.5020	0.3373	0.2593	0.4983	0.3297	0.7154	0.6616
	q	10	10	10	10	10	10	10	10	10
	NMI	0.9531	0.9523	0.9531	0.9523	0.9531	0.9493	0.9121	0.4132	0.1124
	AMI	0.9049	0.9034	0.9049	0.9034	0.9049	0.8999	0.8675	0.3535	0.0466
I-AManPG	Mod.	0.9245	0.8266	0.7306	0.6324	0.5363	0.4367	0.3362	0.1724	0.1100
	purity	1.0000	1.0000	1.0000	1.0000	1.0000	0.9985	0.9704	0.5765	0.3498
	time	1.0201	0.7729	0.6772	0.6584	0.7162	0.7193	1.0411	1.7427	1.5559
	q	17	17	17	17	17	17	17	17	17
	NMI	0.9717	0.9722	0.9665	0.9726	0.9717	0.9709	0.9411	0.4298	0.1183
	AMI	0.9415	0.9424	0.9368	0.9433	0.9415	0.9417	0.9131	0.3733	0.0498
I-AManPG	Mod.	0.9362	0.8382	0.7367	0.6423	0.5427	0.4442	0.3424	0.1721	0.1089
	purity	1.0000	1.0000	0.9947	1.0000	1.0000	0.9989	0.9773	0.5746	0.3379
	time	1.0509	0.9585	0.7564	0.7069	0.8014	0.7777	1.1587	2.0853	1.9374
	q	18	18	18	18	18	18	18	18	18
	NMI	0.9883	0.9883	0.9845	0.9883	0.9796	0.9828	0.9520	0.4339	0.1264
	AMI	0.9753	0.9753	0.9716	0.9753	0.9667	0.9699	0.9365	0.3803	0.0557
I-AManPG	Mod.	0.9453	0.8460	0.7446	0.6474	0.5445	0.4468	0.3427	0.1728	0.1111
	purity	1.0000	1.0000	0.9962	1.0000	0.9914	0.9949	0.9714	0.5634	0.3203
	time	0.9389	0.9057	0.8584	0.9534	0.7417	1.0226	1.0085	2.1798	1.8913
	q	19	19	19	19	19	19	19	19	19
	NMI	1.0000	1.0000	1.0000	1.0000	1.0000	0.9998	0.9600	0.4517	0.1294
	AMI	1.0000	1.0000	1.0000	1.0000	1.0000	0.9998	0.9539	0.4037	0.0563
I-AManPG	Mod.	0.9497	0.8499	0.7503	0.6500	0.5499	0.4498	0.3416	0.1735	0.1113
	purity	1.0000	1.0000	1.0000	1.0000	1.0000	0.9999	0.9679	0.5605	0.3044
	time	0.6577	0.5052	0.6423	0.9890	0.7112	0.4951	1.0755	1.7648	1.7855
	q	20	20	20	20	20	20	20	20	20
	NMI	0.9944	0.9946	0.9946	0.9945	0.9945	0.9927	0.9620	0.4541	0.1322
	AMI	0.9881	0.9885	0.9885	0.9881	0.9883	0.9863	0.9547	0.4056	0.0566
I-AManPG	Mod.	0.9303	0.8338	0.7370	0.6380	0.5402	0.4422	0.3384	0.1731	0.1107
	purity	1.0000	1.0000	1.0000	1.0000	1.0000	0.9985	0.9671	0.5498	0.1751
	time	1.6041	1.2957	1.4677	1.1984	2.4890	1.3366	1.0639	1.9795	2.0597
	q	21	21	21	21	21	21	21	21	21
	NMI	0.9890	0.9890	0.9894	0.9890	0.9887	0.9868	0.9649	0.4649	0.1352
	AMI	0.9764	0.9765	0.9773	0.9765	0.9763	0.9742	0.9500	0.4159	0.0574
I-AManPG	Mod.	0.9124	0.8161	0.7239	0.6263	0.5315	0.4349	0.3358	0.1728	0.1123
	purity	1.0000	1.0000	1.0000	1.0000	0.9996	0.9981	0.9831	0.5642	0.1770
	time	1.7685	1.6543	1.4107	0.8671	1.7232	1.4291	0.9652	2.4757	2.5771
	q	22	22	22	22	22	22	22	22	22
	NMI	0.9835	0.9838	0.9842	0.9834	0.9836	0.9805	0.9532	0.4622	0.1402
	AMI	0.9649	0.9654	0.9663	0.9647	0.9655	0.9618	0.9317	0.4105	0.0606
I-AManPG	Mod.	0.8937	0.8009	0.7114	0.6148	0.5233	0.4278	0.3288	0.1710	0.1120
	purity	1.0000	1.0000	1.0000	1.0000	0.9997	0.9975	0.9784	0.5662	0.1798
	time	1.9044	1.7584	1.6944	1.2244	1.8282	2.0833	1.3244	2.9445	2.5133
	q	23	23	23	23	23	23	23	23	23

and purity of I-AManPG are larger than the results of Danon's algorithm for $q = 18$ and $\mu = 0.2$, Newman's spectral algorithm for $q = 21$ and $\mu = 0.2$ and Newman's spectral algorithm for $q = 19$ and $\mu = 0.3$. NMI, AMI, modularity and purity of I-AManPG are competitive with the results of Louvain's algorithm for $q = 19$ and $\mu = 0.6$ and I-AManPG requires less time.

Table 6.4 also provides promising evidence for the possibility of development of a dynamic adaptation strategy for I-AManPG. When μ_{LFR} takes 0 to 0.7, $q = 10, 17, 18, 19, 20, 21, 22, 23$, i.e., near q_{true}, the modularity, NMI and AMI of I-AManPG decreases as q moves away from q_{true}. Of course this information is not available for the algorithm to use, but it is due to the fact though not reported in Table 6.4 that when $\mu_{\text{LFR}} = 0, 0.1$ the partitionings for $q = 17, 18, 19, 20, 21, 22$ are perfectly nested, i.e., the extra communities of $q + 1$ are refinements of the partition of q by splitting without crossing the ideal community boundaries. When comparing the cases $q = 10$, $q = 17$ and ground-truth, we observe that they all have perfect nesting with respect to each other. From the results of Table 6.4, we observe that when μ_{LFR} takes 0.2 to 0.6, each partitioning for $q = 10, 17, 18, 19, 21, 22, 23$ is well nested with partitioning of q_{true} because the purities are very close to 1.

Real-World Networks. In this section, we apply the I-AManPG algorithm to the four real-world networks mentioned in Section 6.1.1 and compared with the Danon et al.'s algorithm [28], the Louvain method [12] and Newman's spectral optimization method [82]. The results are shown in Table 6.5. From Table 6.5 and Table 6.6, we can analyze the results for each network. For the football network, I-AManPG achieves better NMI, AMI, purity than the other methods and comparable modularity with Louvain method. In addition, we can successfully force Danon, Louvain and Newman's methods to determine assignments with the same number of communities as the ground-truth (12). Note that the computed number of communities is not necessarily the same with 12 even though the modification to the stopping criteria attempts to force this. Even when we force the number of communities to be 12 for the other three methods, we can observe that the partitions from Danon and Newman's methods are not as close as the ground-truth partitions than the partitions from I-AManPG by comparing the results of NMI, AMI and purity. Louvain and the forced Louvian gets the same NMI, AMI and purity results. From Table 6.6, it can be seen that I-AManPG with $q = 10$ has larger purity than the Louvain algorithm, and NMI, AMI are comparable.

For the Karate club network, I-AManPG achieves all 1's for NMI, AMI and purity while the others can not as they can not get the same number of communities as the ground-truth communities. The modularity for I-AManPG is a bit smaller than the others, but it is still comparable to other methods and larger than the forced Newman method which can get the same number of communities. When we try $q = 4$ as the input of I-AManPG in Table 6.6, the NMI, AMI and purity are larger than Danon, Newman, and Louvain algorithms. Modularity in this case is larger than Danon's, and Newman's algorithms and comparable with the Louvain algorithm.

The Polbooks and email network are difficult to analyze as modularity does not predict well the quality of the assignment measured by NMI and AMI. All the results are comparable for these two networks overall. This indicates that the modularity should be replaced with some other cost function. This can be investigated as a future work of this dissertation. Note that, I-AManPG takes a longer time than the other algorithms especially for relatively larger network email network. To improve the efficiency and the adaption on the number of communities, we propose one recursive I-AManPG algorithm in Chapter 7.

6.1.4 Continuation Technique for the Balancing Parameter of λ

As we know, we can apply I-AManPG to solve the community detection problems by solving the following optimization problem

$$\min_{X \in \mathcal{F}_v} f(X) + \lambda ||X||_1, \tag{6.3}$$

where $\lambda > 0$ is a tuning parameter and $\mathcal{F}_{\mathbf{1}_n} = \{X \in \mathbb{R}^{n \times q} : X^T X = I_q, \mathbf{1}_n \in \text{span}(X)\}$ is the domain. $f(X) = -\text{trace}(X^T M X)$ in particular for community detection, where $M = A - A\mathbf{1}_n\mathbf{1}_n^T A/(\mathbf{1}_n^T A \mathbf{1}_n)$ is the modularity matrix and A is the adjacency matrix of the graph.

The choice of the balancing parameter λ is very important to improve the effectiveness for I-AManPG. In this section, we apply the same continuation technique for ARPPG in Algorithm 5 to I-AManPG in Algorithm 17. In this section, we compare the results of I-AManPG algorithm with and without the continuation technique. Let us compare the results on LFR benchmark for n=1000, p=20 and mixing parameter in the range of [0:0.1:0.8] with the tuning parameter λ directly chosen as 0.3 and using the continuation technique in the following algorithm.

There are some choices for the criteria of Algorithm 17. Firstly, we take (the minimum difference of the first two largest magnitude of each row of X) larger than 0.5 to guarantee enough sparse of

Table 6.5: Compare the effectiveness of I-AManPG to other state-of-the-art methods. Each result is an average result of 10 random runs for the real-world networks.

		Football	Karate-2	Polbooks	Email
Danon	NMI	0.7298	0.5305	0.5740	0.5261
	AMI	0.5744	0.3830	0.5075	0.3715
	Mod.	0.5661	0.4087	0.5225	0.4240
	purity	0.9043	0.9412	0.8571	0.8587
	time	0.0269	0.0063	0.0221	2.9917
	q_c	6	4	4	26
Danon_force_q	NMI	0.7031	0.5305	0.5740	0.5531
	AMI	0.5490	0.3830	0.5075	0.3954
	Mod.	0.5183	0.4087	0.5225	0.4188
	purity	0.5826	0.9412	0.8571	0.4119
	time	0.0267	0.0056	0.0215	2.9443
	q_c	12	4	4	42
Louvain	NMI	0.8903	0.5866	0.5745	0.5296
	AMI	0.8208	0.4254	0.5560	0.3763
	Mod.	0.6046	0.4188	0.4986	0.4100
	purity	0.9217	0.9706	0.8476	0.8736
	time	0.0548	0.0353	0.0486	1.9203
	q_c	10	4	3	25
Louvain_force_q	NMI	0.8903	0.5866	0.5745	0.5296
	AMI	0.8208	0.4254	0.5560	0.3763
	Mod.	0.6046	0.4188	0.4986	0.4100
	purity	0.9217	0.9706	0.8476	0.8736
	time	0.0525	0.0278	0.0454	1.6455
	q_c	10	4	3	25
Newman_Eig Sparse	NMI	0.6987	0.6771	0.5201	0.5176
	AMI	0.5611	0.4936	0.4429	0.3805
	Mod.	0.4926	0.3934	0.4672	0.3951
	purity	0.8087	1.0000	0.8476	0.4100
	time	0.0311	0.0082	0.0312	0.3794
	q_c	8	4	4	43
Newman_Eig_force_q Sparse	NMI	0.6987	0.8041	0.5201	0.5176
	AMI	0.5611	0.6618	0.4429	0.3805
	Mod.	0.4926	0.2064	0.4228	0.3950
	purity	0.8087	1.0000	0.8476	0.4100
	time	0.0314	0.0065	0.0291	0.3804
	q_c	8	2	3	42
I-AManPG	NMI	0.9047	1.0000	0.4874	0.5908
	AMI	0.8652	1.0000	0.4756	0.4645
	Mod.	0.5875	0.3715	0.4878	0.2419
	purity	0.9130	1.0000	0.8190	0.5692
	time	0.2214	1.5514	0.0582	13.4378
	q	12	2	3	42

Table 6.6: Compare the effectiveness of I-AManPG with different q which is set to near q_{true}. Each result is an average result of 10 random repeated runs for the real-world networks.

		I-AManPG			
Football	NMI	0.8928	0.9035	0.9047	0.9108
	AMI	0.8194	0.8536	0.8652	0.8633
	Mod.	0.5987	0.6022	0.5875	0.5768
	purity	0.9304	0.9217	0.9130	0.9304
	time	0.2206	0.2449	0.2452	0.0712
	q	10	11	12	13
Karate	NMI	1.0000	0.8041	0.6956	0.5631
	AMI	1.0000	0.6618	0.5147	0.3827
	Mod.	0.3715	0.3922	0.4112	0.3912
	purity	1.0000	1.0000	1.0000	0.9706
	time	1.7951	1.4493	0.4192	0.2041
	q	2	3	4	5
Polbooks	NMI	0.4874	0.4844	0.4853	0.4461
	AMI	0.4756	0.4175	0.3863	0.3331
	Mod.	0.4878	0.4902	0.5121	0.5052
	purity	0.8190	0.8286	0.8381	0.8476
	time	0.1444	0.2353	0.3954	0.3608
	q	3	4	5	6
Email	NMI	0.5497	0.6040	0.6051	0.5908
	AMI	0.4098	0.4775	0.4804	0.4645
	Mod.	0.3422	0.2914	0.2946	0.2419
	purity	0.7095	0.6388	0.6259	0.5692
	time	2.2626	2.6843	6.3719	13.4378
	q	15	25	26	42

Algorithm 17 Algorithm for Assignment Matrix with Continuation

1: $\lambda_0 = \lambda_{max}(M) * q * \rho_0/n$, $\lambda_{max}(M)$ is the largest eigenvalue of the modularity matrix M, ρ_0 is a fixed small number and we set it as 0.5 here;
2: $X_* = \text{I-AManPG}(x_0, \lambda_0)$;
3: $\hat{X}_* = Assign(X_*)$;
4: **while** Some criteria **do**
5: $\quad x_0 = X_*$;
6: $\quad \rho_0 = \rho_0 + 0.01$, $\lambda_0 = \lambda_{max}(M) * q * \rho_0/n$;
7: $\quad X_* = \text{I-AManPG}(x_0, \lambda_0)$;
8: $\quad \hat{X}_* = Assign(X_*)$.
9: **end while**

the result. If we only consider this sparse criteria, the algorithm will update λ for many many times even after $\lambda > 2$. Except that we may also consider the criteria $if\ |||x|-|x_0|||_F/(n*p) > 1e-4, break$ in the while loop to stop it if we have closer results of X_*.

The comparison results are in Table 6.8.From the results in Table 6.8, it is seen that these two ways can get the same NMI, AMI, Modularity and purity results and competitive efficiency for the mixing parameter $\mu = [0 : 0.1 : 0.5]$, when the graph become more noisy, i.e., $\mu = 0.6, 0.7$, the continuation technique can get larger NMI, AMI, Modularity and purity results than choosing λ directly as 0.3 but with more time. When the graph become much more noisy $\mu = 0.7, 0.8$, and it does not have strong community structure to be detected. These two ways behave similarly and continuation technique takes much more time. In addition, we can compute the modularity of the Ground-Truth partition for each LFR benchmark as in Table 6.7. When $\mu = 0.7, 0.8$, the modularity is less than 0.3, and usually we say that there is no community structure.

Table 6.7: Modularity for ground-truth partition for LFR benchmark n=1000, p=20.

	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
Modularity	0.9497	0.8499	0.7503	0.6500	0.5499	0.4499	0.3499	0.2500	0.1501

Whether we use the continuation technique or not depends on our purposes. If one would like to get more precise community structures and regardless of the computational time, then the continuation technique is recommended to be used.

Now, let us compare the results of I-AManPG with the continuation technique to the state-of-the-art methods, Danon's, Newman's, and the Louvain algorithm.From Table 6.9, it is seen that I-AManPG with λ continuation performs much better than Danon and Newman_Eig for $\mu_{LFR} = [0 : 0.1, 0.6]$ in terms of both quality of assignment and time. For the very noisy cases $\mu_{LFR} = 0.7, 0.8$, all algorithms produce far less effective assignments. However, I-AManPG and Louvain algorithm are relatively more effective than the other two algorithms.

When we compare the effectiveness with Louvain algorithm, it is seen that I-AManPG with continuation can get the same results of NMI, AMI, Modularity and purity for $\mu_{LFR} = [0 : 0.1, 0.4]$. For $\mu_{LFR} = 0.5, 0.6, 0.7\ and\ 0.8$, I-AManPG can get larger NMI, AMI and competitive modularity and purity compared with Louvain algorithm. From the comparison of efficiency with Louvain

Table 6.8: Compare the results of I-AManPG by choosing the balance parameter λ directly and using the continuation technique. An average result of 10 random runs for the randomly generated graphs by LFR benchmark networks.

		μ_{LFR}								
		0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
I-AManPG-λ-directly	NMI	1.0000	1.0000	1.0000	1.0000	1.0000	0.9998	0.9600	0.4517	0.1294
	AMI	1.0000	1.0000	1.0000	1.0000	1.0000	0.9998	0.9539	0.4037	0.0563
	Mod.	0.9497	0.8499	0.7503	0.6500	0.5499	0.4498	0.3416	0.1735	0.1113
	purity	1.0000	1.0000	1.0000	1.0000	1.0000	0.9999	0.9679	0.5605	0.3044
	time	0.6103	0.4981	0.6764	0.9192	0.6559	0.4580	1.0048	1.5865	1.6467
I-AManPG-λ-continuation	NMI	1.0000	1.0000	1.0000	1.0000	1.0000	0.9998	0.9824	0.5684	0.1212
	AMI	1.0000	1.0000	1.0000	1.0000	1.0000	0.9998	0.9811	0.5357	0.0567
	Mod.	0.9497	0.8499	0.7503	0.6500	0.5499	0.4498	0.3475	0.1951	0.1312
	purity	1.0000	1.0000	1.0000	1.0000	1.0000	0.9999	0.9891	0.6721	0.1654
	time	1.5690	1.3803	1.1894	1.3832	1.1236	1.2255	3.3668	34.9186	42.8637
	λ	0.2534	0.2291	0.2052	0.1818	0.1591	0.1367	0.1184	0.1041	0.0999
	# updates of λ	2.0000	2.0000	2.0000	2.0000	2.0000	2.0000	2.0000	2.0000	2.3000

algorithm, I-AManPG can obtain the competitive time for $\mu_{LFR} = [0 : 0.1 : 0.6]$ and is slower than Louvain algorithm for $\mu_{LFR} = 0.7, 0.8$.

For the criteria of Algorithm 17, it is more reasonable to change the distance criteria (the minimum difference of the first two largest magnitude of each row of X) larger than 0.5 to the ratio criteria (the minimum ratio of the first two largest magnitude of each row of X) larger than 5 to guarantee enough sparse of the result. If we only consider this sparse criteria, the algorithm will update λ for many many times even after $\lambda > 2$. Except that we may also consider the criteria $if\ |||x| - |x_0|||_F/(n * p) > 1e-6, break$ in the while loop to stop it if we have closer results of X_*. In addition, we add one more criteria $if\ \rho > 1, break$.

From the results in Table 6.10, we can find that the ratio criteria is also a choice for the criteria. I-AManPG with continuation technique with ratio criteria can get the same NMI, AMI, Modularity and purity results as I-AManPG with continuation technique with distance criteria. For the case $\mu = 0$, I-AManPG with continuation technique with ratio criteria takes less time than I-AManPG with continuation technique with distance criteria. For the other cases, I-AManPG with continuation technique with ratio criteria takes more time than I-AManPG with continuation technique with distance criteria.

Table 6.9: Compare the effectiveness of I-AManPG with λ continuation to other state-of-the-art methods. Each result is an average result of 10 random runs for the graphs generated randomly as LFR benchmark networks. q_c is the computed number of communities, q is the input parameter for I-AManPG and "force_q" algorithms denote the algorithms by adding a forcing criteria such that the algorithms stop when q_c is smaller than or equal to $q_{true} = 20$.

		μ_{LFR}								
		0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
Danon	NMI	0.9998	0.9891	0.9394	0.8504	0.7331	0.5808	0.3781	0.1412	0.0548
	AMI	0.9998	0.9870	0.9166	0.7922	0.6399	0.4736	0.2878	0.0935	0.0215
	Mod.	0.9496	0.8436	0.7225	0.5920	0.4687	0.3452	0.2458	0.1892	0.1814
	purity	0.9999	0.9938	0.9739	0.9414	0.9058	0.8344	0.6886	0.4269	0.3095
	time	2.7403	2.8018	2.7810	2.8066	2.7341	2.7161	2.7317	2.7749	2.7852
	q_c	20	20	18	15	11	9	7	7	8
Danon_force_q	NMI	0.9998	0.9889	0.9397	0.8494	0.7309	0.5794	0.3939	0.1778	0.0954
	AMI	0.9998	0.9870	0.9201	0.7962	0.6415	0.4727	0.2950	0.1035	0.0282
	Mod.	0.9496	0.8433	0.7209	0.5868	0.4615	0.3394	0.2431	0.1872	0.1794
	purity	0.9999	0.9935	0.9714	0.9349	0.8954	0.8198	0.6778	0.4201	0.3065
	time	2.7430	2.7314	2.7501	2.7762	2.7314	2.7514	2.7580	2.7477	2.7608
	q_c	20	20	20	20	20	20	20	20	20
Louvain	NMI	1.0000	1.0000	1.0000	1.0000	1.0000	0.9987	0.9805	0.2862	0.0784
	AMI	1.0000	1.0000	1.0000	1.0000	1.0000	0.9974	0.9652	0.2249	0.0358
	Mod.	0.9497	0.8499	0.7503	0.6500	0.5499	0.4496	0.3477	0.2098	0.1967
	purity	1.0000	1.0000	1.0000	1.0000	1.0000	0.9999	0.9950	0.4734	0.2660
	time	0.5346	0.6764	1.2988	1.8051	2.5857	3.2167	4.3215	8.5401	8.5827
	q_c	20	20	20	20	20	20	19	11	11
Louvain_force_q	NMI	1.0000	1.0000	1.0000	1.0000	1.0000	0.9987	0.9805	0.2981	0.0847
	AMI	1.0000	1.0000	1.0000	1.0000	1.0000	0.9974	0.9652	0.2382	0.0392
	Mod.	0.9497	0.8499	0.7503	0.6500	0.5499	0.4496	0.3477	0.2098	0.1967
	purity	1.0000	1.0000	1.0000	1.0000	1.0000	0.9999	0.9950	0.4642	0.2418
	sum_time	0.5733	0.7016	1.0253	1.2292	1.6447	1.9749	2.6044	5.1309	5.1496
	q_c	20	20	20	20	20	20	19	12	12
Newman_Eig Sparse	NMI	1.0000	0.6251	0.8797	0.7618	0.5400	0.4247	0.2852	0.1265	0.0516
	AMI	1.0000	0.5101	0.8613	0.7127	0.4707	0.3377	0.2116	0.0846	0.0244
	Mod.	0.9498	0.5271	0.6391	0.4721	0.3011	0.2638	0.1955	0.1523	0.1426
	purity	1.0000	0.8230	0.9120	0.8080	0.6470	0.6590	0.5710	0.4140	0.3490
	time	0.6997	0.3057	0.4958	0.4621	0.4470	0.3238	0.2723	0.2773	0.2716
	q_c	20	11	23	20	15	8	6	6	6
Newman_Eig_force_q Sparse	NMI	1.0000	0.6251	0.8501	0.7618	0.5400	0.4247	0.2852	0.1265	0.0516
	AMI	1.0000	0.5101	0.8233	0.7127	0.4707	0.3377	0.2116	0.0846	0.0244
	Mod.	0.9498	0.5271	0.5396	0.4721	0.3011	0.2638	0.1955	0.1523	0.1426
	purity	1.0000	0.8230	0.8920	0.8080	0.6470	0.6590	0.5710	0.4140	0.3490
	time	0.5973	0.2935	0.4171	0.4340	0.4191	0.3101	0.2692	0.2744	0.2763
	q_c	20	11	20	20	15	8	6	6	6
I-AManPG-λ-continuation	NMI	1.0000	1.0000	1.0000	1.0000	1.0000	0.9998	0.9824	0.5684	0.1212
	AMI	1.0000	1.0000	1.0000	1.0000	1.0000	0.9998	0.9811	0.5357	0.0567
	Mod.	0.9497	0.8499	0.7503	0.6500	0.5499	0.4498	0.3475	0.1951	0.1312
	purity	1.0000	1.0000	1.0000	1.0000	1.0000	0.9999	0.9891	0.6721	0.1654
	sum_time	1.5690	1.3803	1.1894	1.3832	1.1236	1.2255	3.3668	34.9186	42.8637
	λ	0.2534	0.2291	0.2052	0.1818	0.1591	0.1367	0.1184	0.1041	0.0999
	# updates of λ	2.0000	2.0000	2.0000	2.0000	2.0000	2.0000	2.0000	2.0000	2.3000

Table 6.10: Compare the results of I-AManPG by choosing the balance parameter λ directly and using the continuation technique with ratio criteria and $\rho \leq 1$. An average result of 10 random runs for the randomly generated graphs by LFR benchmark networks.

		μ_{LFR}								
		0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
I-AManPG-λ-directly	NMI	1.0000	1.0000	1.0000	1.0000	1.0000	0.9998	0.9600	0.4517	0.1294
	AMI	1.0000	1.0000	1.0000	1.0000	1.0000	0.9998	0.9539	0.4037	0.0563
	Mod.	0.9497	0.8499	0.7503	0.6500	0.5499	0.4498	0.3416	0.1735	0.1113
	purity	1.0000	1.0000	1.0000	1.0000	1.0000	0.9999	0.9679	0.5605	0.3044
	time	0.6381	0.4846	0.6009	0.9273	0.6475	0.4750	1.0690	1.5725	1.6410
I-AManPG-λ-continuation	NMI	1.0000	1.0000	1.0000	1.0000	1.0000	0.9998	0.9841	0.5836	0.1253
	AMI	1.0000	1.0000	1.0000	1.0000	1.0000	0.9998	0.9829	0.5521	0.0608
	Mod.	0.9497	0.8499	0.7503	0.6500	0.5499	0.4498	0.3478	0.2003	0.1404
	purity	1.0000	1.0000	1.0000	1.0000	1.0000	0.9999	0.9902	0.6868	0.1722
	time	0.5395	6.7830	5.1047	6.8252	6.4411	4.9513	7.6709	122.3704	174.9319
	λ	0.2485	0.4058	0.3822	0.3387	0.2963	0.2547	0.2205	0.1940	0.1850
	# updates of λ	1.0000	9.1000	10.0000	10.0000	10.0000	10.0000	10.0000	10.0000	10.0000

We also implement I-AManPG with and without continuation technique on the real-world networks, the Football, Karate Club, and Polbooks networks. Here, we take the ratio criteria as one example to compare the results in Table 6.11. For Karate club network and the Polbooks network, the effectiveness of I-AManPG is not improved by using the λ continuation technique. For the Football network, it can be seen that I-AManPG with the λ continuation technique can improve the effectiveness of I-AManPG. NMI, AMI, modularity and purity by I-AManPG with λ continuation technique are larger than I-AManPG with λ choosing directly while the former takes longer time than the latter. If one would like to get more precise community structures and regardless of the computational time, then the λ continuation technique is recommended.

The choice of the balancing parameter λ plays an important role for clustering. The continuation technique is a good idea to choose λ. As mentioned before, the criteria in the continuation technique Algorithm 17 can be chosen as well. Another criteria is to use the cost function by modifying this criteria to some other criteria like $modu(\hat{X}_*) > modu(\hat{x}_0)$. This can be taken as a future work of this dissertation.

Table 6.11: Compare the results of I-AManPG by choosing the balance parameter λ directly and using the continuation technique with ratio criteria and $\rho \leq 1$. An average result of 10 random runs for the randomly generated graphs by real-world networks.

		Football	Karate	Polbooks
I-AManPG-λ-directly	NMI	0.9047	1.0000	0.4874
	AMI	0.8652	1.0000	0.4756
	Mod.	0.5875	0.3715	0.4878
	purity	0.9130	1.0000	0.8286
	time	0.2778	2.3722	0.1529
	q	12	2	3
I-AManPG-λ-continuation	NMI	0.9242	1.0000	0.4874
	AMI	0.8979	1.0000	0.4756
	Mod.	0.6005	0.3715	0.4878
	purity	0.9217	1.0000	0.8286
	time	21.5983	21.8246	1.9287
	q	12	2	3
	λ	0.9218	0.2781	0.3167
	# updates of λ	10.0000	10.0000	10.0000

6.2 Inexact Accelerated Manifold Proximal Gradient Optimization for the k-means model

6.2.1 k-means Model

The classic k-means model is commonly written in terms of k centroids. It can also be written in terms of assignment matrices. Let $A \in \mathbb{R}^{n\times d}$ be a data matrix where each data point in $\mathbb{R}^d$ corresponds to a row of A. Let $X \in \mathbb{R}^{n\times q}$ be the assignment matrix such that $X^TX = I_q, XX^T\mathbf{1}_n = \mathbf{1}_n$, which forces non-zero elements in each column of X to have the same value (i.e., $\frac{1}{\sqrt{n_i}}$) so X would be the normalized assignment matrix. The classic k-means model can also be rewritten as [23]

$$\min_X ||A - XX^TA||_F^2 \quad \text{s.t. } X \in \mathcal{F}_{\mathbf{1}_n} = \{X \in \mathbb{R}^{n\times q} : X^TX = I_q, XX^T\mathbf{1}_n = \mathbf{1}_n\}. \tag{6.4}$$

To see this, we notice that $||A - XX^TA||_F^2 = \sum_{i=1}^n ||A_{(i)} - X_{(i)}X^TA||_2^2$, where $A_{(i)}$ and $X_{(i)}$ denote the i-th row of A and X, respectively, and $X_{(i)}X^TA$ denotes the centroid of the cluster the i-th data point belongs to [15].

6.2.2 The Connection Between k-means Model and AManPG

To deal with k-means model, that is to minimize the cost function $||A - XX^TA||_F^2$, where $X \in \mathcal{F}_{\mathbf{1}_n} = \{X \in \mathbb{R}^{n\times q} : X^TX = I_q, XX^T\mathbf{1}_n = \mathbf{1}_n\}$.

Minimizing $||A-XX^TA||_F^2$ is equivalent to minimizing $-\operatorname{tr}(X^TAA^TX)$ over $\mathcal{F}_{\mathbf{1}_n}$, also we know that X is sparse, so we can add a penalty term $\lambda||X||_1 := \lambda\sum_{ij}|X_{ij}|$ as we did for community detection, where $\lambda > 0$ is a tuning parameter controlling the balance between variance and sparsity.

Therefore, to deal with k-means model, we just need to solve a constrained Stiefel optimization problem

$$X_* = \underset{X\in\mathcal{F}_{\mathbf{1}_n}}{\operatorname{argmin}} -\operatorname{tr}(X^TAA^TX) + \lambda\|X\|_1, \tag{6.5}$$

where$A \in \mathbb{R}^{n\times d}$ be a data matrix, and X_* is the optimal assignment matrix.

We have shown that the feasible set $\mathcal{F}_{\mathbf{1}_n}$ is a submanifold in Section 4.1, so we can apply I-AManPG algorithm to the optimization problem (6.5). So, I-AManPG over the feasible set can be applied to k-means model.

6.2.3 Numerical Experiments

To see the performance of AManPG method on k-means model, we compare the results with the classical k-means clustering method (k-means++) on some datasets. The datasets that we use are listed as follows. The first 3 datasets are obtained from UCI Machine Learning Repository [4] which is a good data source. The last dataset is obtained from ELKI.

- The Zoo dataset is obtained from UCI and it contains 101 animals with 16 attributes and these animals can be classified into 7 classes.
- Iris dataset is one famous clustering dataset from Fisher [38], and it contains 150 iris plants with 4 attributes. It can be classified into 3 classes of 50 instances each, where each class refers to a type of iris plant.
- The leaf dataset [104] comprises 340 samples from 30 different plant species, and each plant comprises 14 shape and texture attributes.
- Mouse Head dataset is an artificial data set from ELKI. The original mouse head dataset contains 500 data points including 10 noise data points. Here we do not consider the noises and only consider the other 490 data points. So the mouse head dataset used in this dissertation contains 490 data points with 2 attributes which can be classified into 3 classes, which refer to head, left ear and right ear.

For the classical k-means method we use the k-means clustering method with k-means++ as the initiation algorithm. For I-AManPG method, we use the same initial assignment as in k-means clustering and set the balancing parameter directly as $maxeig(AA^T) * q/n$. We call the function $idx = kmeans(X, k)$ for k-means method in Matlab, and this function uses the k-means++ algorithm as the cluster center initialization by default.This function is implemented in an C++ toolbox, while the implementation of I-AManPG is just the simple Matlab implementation. As a result, we are mainly concerned with evaluating the quality of results of I-AManPG relative to that of kmeans++ for Euclidean data. This shows the flexibility of the approach, i.e., graph-based problems and Euclidean data-based problems. The computational time is also reported to motivate further development of the I-AManPG algorithm to improve efficiency before implementing an efficient high-performance codes in the C++ ROPTLIB library [52].

Table 6.12: Performance of I-AManPG to k-means model over 10 runs

Datasets	size	q	Measurements	I-AManPG	k-Means++
Iris	150×4	3	NMI	0.7132	0.7330
			AMI	0.6837	0.7187
			purity	0.8887	0.8860
			time(s)	0.1061	0.0063
Mouse	490×2	3	NMI	0.7446	0.6173
			AMI	0.7240	0.5824
			purity	0.9163	0.8388
			time(s)	0.5923	0.1535
Zoo	101×16	7	NMI	0.6332	0.6353
			AMI	0.5715	0.5653
			purity	0.7109	0.6990
			time(s)	0.2801	0.0126
Leaf	340×14	30	NMI	0.6299	0.6420
			AMI	0.4376	0.4558
			purity	0.5841	0.5826
			time(s)	0.4669	0.0075

The comparison results are shown in Table 6.12 and Figure 6.2. The NMI, AMI, and purity are used as the measurements for the quality of the clustering assignments. The computational time is included for completeness. From the results in Table 6.12 and Figure 6.2, it can be seen that the I-AManPG algorithm can get a better clusters assignment of the mouse head dataset than the k-means clustering algorithm with k-means++ initialization algorithm from the NMI, AMI, and

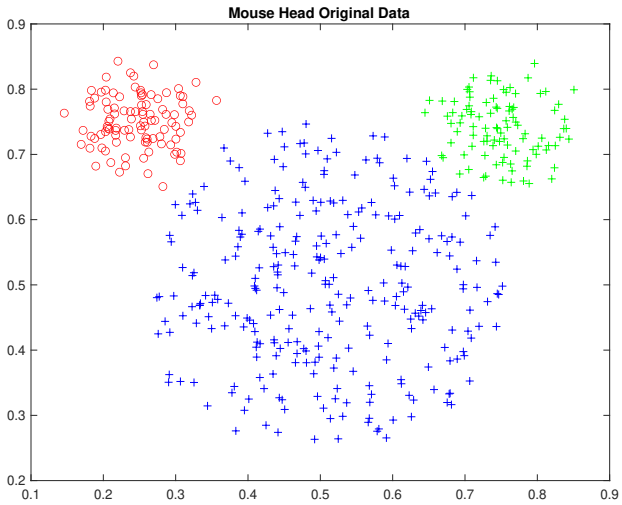
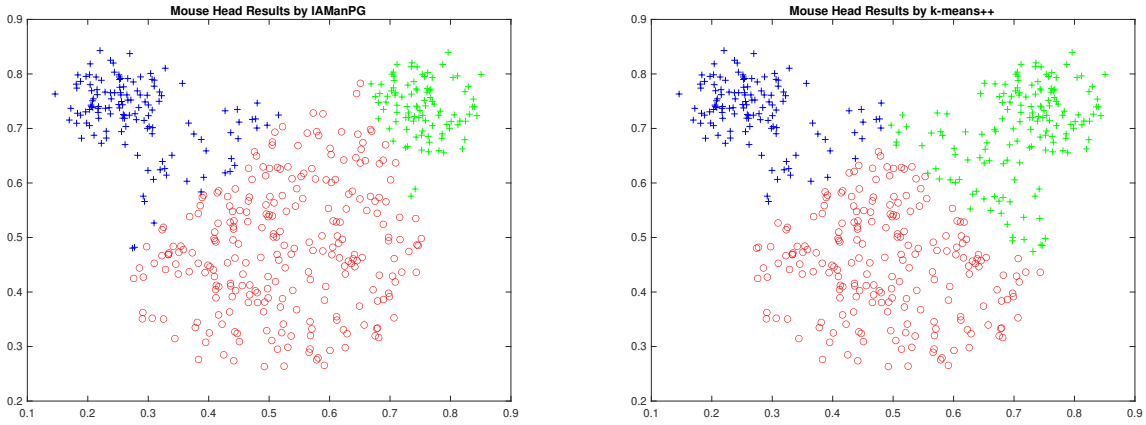


Figure 6.1: Mouse Head Original Data



(a) Mouse Head Results by IAManPG

(b) Mouse Head Results by k-means++

Figure 6.2: Mouse Head results

purity. For the Iris, Zoo, and Leaf datasets, the I-AManPG algorithm can get comparable NMI, AMI, and purity as the k-means clustering algorithm with k-means++ initialization algorithm in terms of the effectiveness.

6.3 Inexact Accelerated Manifold Proximal Gradient Optimization for Discriminative k-means model

A second more complicated k-means related model on Euclidean data is often motivated by the assumption of high dimensionality of the data or the assumption that the, high or possibly moderate, dimensionality is an over-parameterization of the data that admits a representation in a space of smaller dimension. Here we are interested in the problem as an increased complexity problem that subsumes the k-means model that can be handled by the Riemannian approach of I-AManPG. In [33], the dimensionality reduction technique, Linear Discriminant Analysis, is combined with the k-means clustering. It is further analyzed in [121] and is shown to be equivalent to a kernel k-means. The resulting optimization problem is given therein by

$$\min_{X \in \mathcal{A}_{n\times q}, \lambda_{regu}>0} \operatorname{trace}(X^T(I_n + \frac{1}{\lambda_{regu}} A^T A)^{-1} X) + \log\det(I_n + \frac{1}{\lambda_{regu}} A^T A), \tag{6.6}$$

where A is the data matrix, $\lambda_{regu} > 0$ is the regularization parameter. This problem is solved [33] by alternating between the computation of X for a given λ_{regu} and the computation of λ_{regu} for a given X. Given λ_{regu}, (6.6) is in the form of (1.5). This can be reformulated into

$$\min_{X \in \mathcal{A}_{n\times q}, \lambda_{regu}>0} \operatorname{trace}(X^T(I_n + \frac{1}{\lambda_{regu}} A^T A)^{-1} X) + \log\det(I_n + \frac{1}{\lambda_{regu}} A^T A) + \lambda||X||_1, \tag{6.7}$$

which is in form of (1.10). Given λ_{regu}, we apply the I-AManPG algorithm to the discriminative k-means problem (6.7).

We compare the results of I-AManPG with some classic algorithms, the DisKmeans algorithm [121], Local Linear Embedding (LLE) [99], and Laplacian Eigenmap (LEI) [10]. These algorithms are described in Section 2.3.2.

6.3.1 Numerical Experiments

To compare the performance of I-AManPG with some representative clustering algorithms on discriminative k-means, we use the following 4 datasets. The first 3 datasets are obtained from UCI

Machine Learning Repository, and the other one is USPS dataset from (ftp://ftp.kyb.tuebingen.mpg.de/pub/bs/data/).

- Segment dataset has 2309 instances which were drawn randomly from a database of 7 outdoor images. Each instance has 19 attributes.
- Pendigits dataset is a handwritten digit classification dataset [3]. It contains 10992 handwritten digits $[0, 1, ..., 9]$, where each handwritten digit is made up of the x and y coordinates of the pen traced across a digital screen. Each instance has 16 attributes.
- Satimage dataset consists of the multi-spectral values of pixels in 3×3 neighborhoods in a satellite image. There are 36 attributes (=4 spectral bands $\times$ 9 pixels in neighborhood). There are 6435 samples, each sample is categorized as one of 6 classes.
- USPS dataset is a digit dataset automatically scanned from envelopes by the U.S. Postal Service containing a total of 9298 16×16 pixel grayscale samples, (256 attributes) [55]. These samples are categorized into 10 groups, where each group was drawn from a different ZIP Code region.

A summary of the data sets used for discriminative k-means model is given in Table 6.13.

Table 6.13: Summary of data sets used for discriminative k-means model

Data Set	# INST (n)	# DIM (d)	# CL (q_{true})
segment	2309	19	7
pendigits	10992	16	10
satimage	6435	36	6
USPS	9298	256	10

To make the results of different algorithms comparable, we first run k-means and the clustering result of k-means is used to construct the set of k initial centroids, for all experiments. To measure the performance of the algorithms, we use the NMI and Accuracy (ACC) in (6.8) as two measurements for clustering quality as the authors did in [121]. Since the codes evaluated in [121] are not publicly available, we only compare the clustering quality and do not compare the computation time. As with the k-means model discussion, we are concerned with evaluating the quality of results of I-AManPG relative to state-of-the-art algorithms for Euclidean data to demonstrate the flexibility of the method and its viability as the candidate of an efficient library version.

For n samples in a dataset, let y_i be the class label for the i-th sample and $\hat{y}_i$ the predicted value. Accuracy between the class labels y and the predicted values $\hat{y}$ for the clustering tasks is

defined by

$$accuracy(y, \hat{y}) = \max_{perm \in P} \frac{1}{n} \sum_{i=0}^{n-1} \mathbf{1}(perm(\hat{y}_i = y_i)), \tag{6.8}$$

where $\mathbf{1}(\hat{y}_i = y_i)$ is the indicator function, and$P$ is the set of all permutations in $[1 : q]$, where q is the number of clusters. It is important to note that there are $q!$ permutations in P, which is quite high and the computation of accuracy is therefore prohibitive if we apply blindly this formula. The Hungarian algorithm [63, 58] is used to compute it in $O(q^3)$ which is significantly faster.

We use $q = q_{true}$ which is listed in the last column of Table 6.13 as the input for I-AManPG. In addition, I-AManPG's another two parameters, the balancing parameter λ and the regularization parameter λ_{regu} are set as follows. The balancing parameter λ as $\lambda = (maxeig((I_n + \frac{1}{\lambda_{regu}} A^T A)^{-1}) * q + \log\det(I_n + \frac{1}{\lambda_{regu}} A^T A))/n$ by taking the second half cost function the regularization term $\log\det(I_n + \frac{1}{\lambda_{regu}} A^T A)$ into account, as the balancing parameter is used to make the results as sparse as possible. The choice of λ_{regu} follows the optimal choice which is shown in Figure 1 of [121].

Table 6.14: Numerical Results on the DisKmeans Model. DisKmeans stands for DisKmeans algorithm in [121]. LLE stands for Local Linear Embedding and LEI for Laplacian Eigenmap.NA stands for Not Applicable.

Dataset	Method	λ	λ_{regu}	NMI	ACC
segment	I-AManPG	0.0704	1	0.605	0.422
	DisKmeans	NA	1	0.628	0.687
	LLE	NA	NA	0.539	0.594
	LEI	NA	NA	0.618	0.663
pendigits	I-AManPG	0.0199	10	0.663	0.665
	DisKmeans	NA	10	0.669	0.699
	LLE	NA	NA	0.577	0.599
	LEI	NA	NA	0.645	0.697
satimage	I-AManPG	0.0289	1e3	0.598	0.699
	DisKmeans	NA	1e3	0.593	0.701
	LLE	NA	NA	0.493	0.627
	LEI	NA	NA	0.548	0.663
usps	I-AManPG	0.0123	1e3	0.645	0.679
	DisKmeans	NA	1e3	0.647	0.712
	LLE	NA	NA	0.569	0.631
	LEI	NA	NA	0.640	0.700

The results for the DisKmeans model are in Table 6.14. For the segment dataset, the NMI by I-AManPG is larger than the LLE, and less than but close to the LEI and DisKmeans algorithm. The ACC by I-AManPG is less than the other algorithms. For the pendigits network, the NMI and ACC by I-AManPG are larger than the LLE algorithm and close to DisKmeans, and LEI. For the satimage dataset, I-AManPG performs relatively better than the others overall. NMI by I-AManPG is higher than all the other algorithms. ACC by I-AManPG is larger than LLE, and LEI and close to the DisKmeans algorithm. For the USPS dataset, NMI and ACC by I-AManPG is larger than LLE and close to DisKmeans and LEI algorithm. Overall, the performance of IAManPG are comparable to the other algorithms in terms of NMI and ACC. The I-AManPG algorithm provides some new opportunities for solving the clustering problems in terms of Riemanian Optimization.

6.4 Inexact Accelerated Manifold Proximal Gradient Optimization for Normalized Cut

Normalized cut has been widely used for image segmentation. Its optimization formulation is given by

$$\min_{X \in \mathcal{A}_v^{n,q}} f_{\mathrm{NC}}(X) = -\mathrm{trace}(X^T D^{-1/2} W D^{-1/2} X). \tag{6.9}$$

This problem assumes that graph-based data are represented by an appropriate matrix characterizing the relationships between the basic data elements from the application problem. In the case of gray image segmentation, the matrix $W \in \mathbb{R}^{mn \times mn}$ is an affinity matrix of an m by n pixels gray image, $D \in \mathbb{R}^{mn \times mn}$ is a diagonal matrix with $D_{ii} = \sum_{j=1}^{mn} W_{ij}$, and $v = \mathrm{diag}(D^{1/2})$. Here, we use the approach in [105] to choose W and D, to further shift the diagonal entries of W and D by a constant.

As mentioned in Section 2.4.1,(6.9) can be optimized by the weighted kernel k-means algorithm, see e.g., [31, Algorithm 1]. Since that Problem (6.9) has many low-quality local minimizers and descent algorithms are usually not able to escape from them, the initialization plays an important role in finding an acceptable solution. Four initialization methods including Bach and Jordan [6], Shi and Malik [103], Karypis and Kumar [60], and the proposed one based on I-AManPG are described in Section 2.4.2 and the performance of these initialization methods are compared in this section.

Let U be the $n \times q$ matrix of the q leading eigenvectors of the matrix $D^{-1/2}WD^{-1/2}$. If X is only required to be orthonormal, then U is a global minimizer of (6.9). Since U is unlikely to be in $\mathcal{A}_v^{n,q}$, one approach is to find a matrix in $\mathcal{A}_v^{n,q}$ that is close to U. Different notions of closeness yield different methods. Bach and Jordan [6] seek to find a matrix $Y \in \mathcal{A}_v^{n,q}$ that minimizes

$$\|UU^T - YY^T\|_F. \tag{6.10}$$

The difference between U and Y is measured by the difference between the two orthogonal projection matrices. The weighted kernel k-means is suggested to solve (6.10) see [6, Figure 1]. However, similar to (6.9), the kernel k-means for (6.10) may also get stuck in a local minimizer. We use k-means++ as implemented in Matlab for the initialization of the kernel k-means for (6.10).

The task of Shi and Malik [103] is to find an indicator matrix Z and a q-by-q orthonormal matrix Q that minimize

$$\|Z - \tilde{U}Q\|_F.$$

It uses the alternate minimization algorithm to find Z and Q in $\|Z-\tilde{U}Q\|_F$. Note that this approach neither guarantees to find the global optimum nor use the weight vector v. Therefore, this approach may not find a satisfactory solution. Here, we use the C and Matlab hybrid implementation from [105].

Karypis and Kumar [60] give a fast, multi-level graph partitioning algorithm that produces equally-sized clusters and is called METIS. It is shown to be an effective method for the kernel k-means initialization. Note that METIS does not aim to minimize the objective (6.9). To implement Karypis and Kumar [60], we use the C implementation from `http://glaros.dtc.umn.edu/gkhome/metis/metis/download` with the Matlab interface from `https://github.com/dgleich/metismex`.

To get the initial X_0 for the weighted kernel k-means algorithm to solve (6.9), we propose to use some number of iterations of I-AManPG on related and equivalent problem (6.11).

Specifically, Problem (6.9) can be reformulated as

$$\min_{X \in \mathcal{F}_v} -\text{trace}(X^T D^{-1/2} W D^{-1/2} X) + \lambda \|X\|_1, \tag{6.11}$$

which can be optimized by I-AManPG. We further propose to gradually increasing λ rather than a fixed value of λ since increasing λ tends to give better solutions in our experiments[2]. The clusters

[2]The λ in I-AManPG increases by 0.01, 0.04, and 0.2

are specified by $P_{\mathcal{A}_v^{n,q}}(X_*) = \text{diag}(v)P_{\mathcal{A}_{1_n}^{n,q}}(\text{diag}(v)^{-1}X)$, where $P_{\mathcal{A}_{1_n}^{n,q}}(X) = \left[\frac{b_1}{\|b_1\|} \quad \frac{b_2}{\|b_2\|} \quad \cdots \quad \frac{b_q}{\|b_q\|}\right]$, $b_j \in \mathbb{R}^n$ for $j = 1, 2, \ldots, q$, and

$$(b_j)_i = \begin{cases} 1 & \text{if } X_{ij} \text{ has the largest magnitude in the } i\text{-th row;}^3 \\ 0 & \text{otherwise.} \end{cases}$$

Such clusters are then used as initializations for the weighted kernel k-means algorithm. The initialization by I-AManPG, essentially uses a normalized cut problem to start a state-of-the-art method for the same normalized cut problem. At first glance this seems difficult to motivate. Note however, that we exploit two key properties of the Riemannian approach in this dissertation. The feasible set is changed to $\mathcal{F}_v$ to exploit the geometry rigorously. The cost function is changed to combine a trace term and a sparsity term. When optimized alone the trace term is driven by eigenvalue and invariant subspace information in a manner similar to the other standard initialization approaches. As a result, standard descent or related methods could experience the pull of unacceptable local minima and the associated subspaces. The sparsity term forces the algorithm to move toward vectors that satisfy the demand of the problem for a minimizing assignment matrix. The weight λ allows the tuning of this modifcation to the path produced in $\mathcal{F}_v$. Of course, if the sparsity term alone was optimized it would also tend to fail to solve the problem since it would not necessarily satisfy the spectral properties demanded by the problem from the trace term or the cost functions of the other standard initialization algorithms. This motivates the expectation of superior initialization from the Riemannian I-AManPG approach.

The above four initialization methods are respectively denoted by BJ, SM, ME, and AM. Their combinations with the weighted kernel k-means algorithms are respectively denoted by BJ-k, SM-k, ME-k, and AM-k. The implementation of the weighted kernel k-means algorithm is modified from [24][4]. The tested 12 images are from [105] and the built-in images in Matlab. We further resize them to have 160-by-160 pixels as shown in Figure 6.3. The 12 images are baby, cameraman, coins, football, gantrycrane, liftingbody,onion, panther, pears, peppers, saturn, and tape images. We implement the four initialization methods and then combine with the weighted kernel k-means algorithm on each image with 3, 5, and 7 clusters.

As an example, the segmentation results of the four methods for the baby image are shown in Figure 6.4. It can be seen that the segmentation results by AM-k for 3, 5, and 7 clusters perform better than BJ-k and ME-k methods and comparable with SM-k method.

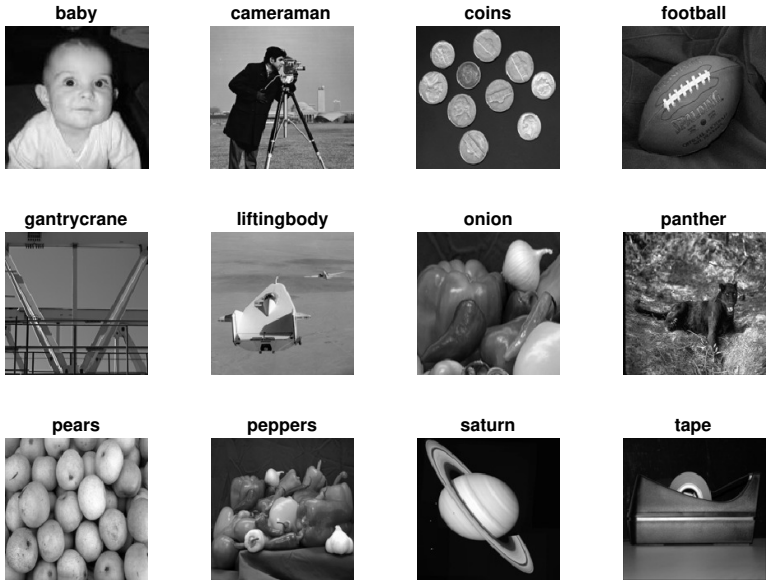


Figure 6.3: The tested images

An average of the negative function values $-f_{\mathrm{NC}}$ of 10 random runs are reported in Figure 6.5 and Figure 6.6.We do not report the computational time since the implementations of these methods use different languages and their computational time can not be rigorously compared. The qualities of these methods are compared based on the object function value f_{NC}. As shown in the figures, METIS initializations are not preferred since they do not aim to minimize f_{NC}. Though SM, SM-k, BJ, BJ-k are competitive to AM and AM-k in many cases, they do not perform well on certain images, such as SM and SM-k for the football image and and the tape image with 3 clusters, and BJ and BJ-k for the pears, the tape image with 3 clusters. I-AManPG based methods are clearly most robust in the sense of minimizing the function f_{NC}, where the function f_{NC} is minimized over $\mathcal{F}_v$ and then projects the final solution onto $\mathcal{A}_v^{n,q}$. The values of $-f_{\mathrm{NC}}$ by AM-k are often the highest one. Even if they are not, they are still close to the highest ones. The empirical evidence supports the expectations given in the motivation discussion above that I-AManPG is competitive with or superior to initialization strategies in the current literature.

[4]The implementation in [24] is for unweighted kernel k-means. We modified it for weighted kernel k-means.

Figure 6.4: Image Segmentation Comparisons on Baby Image

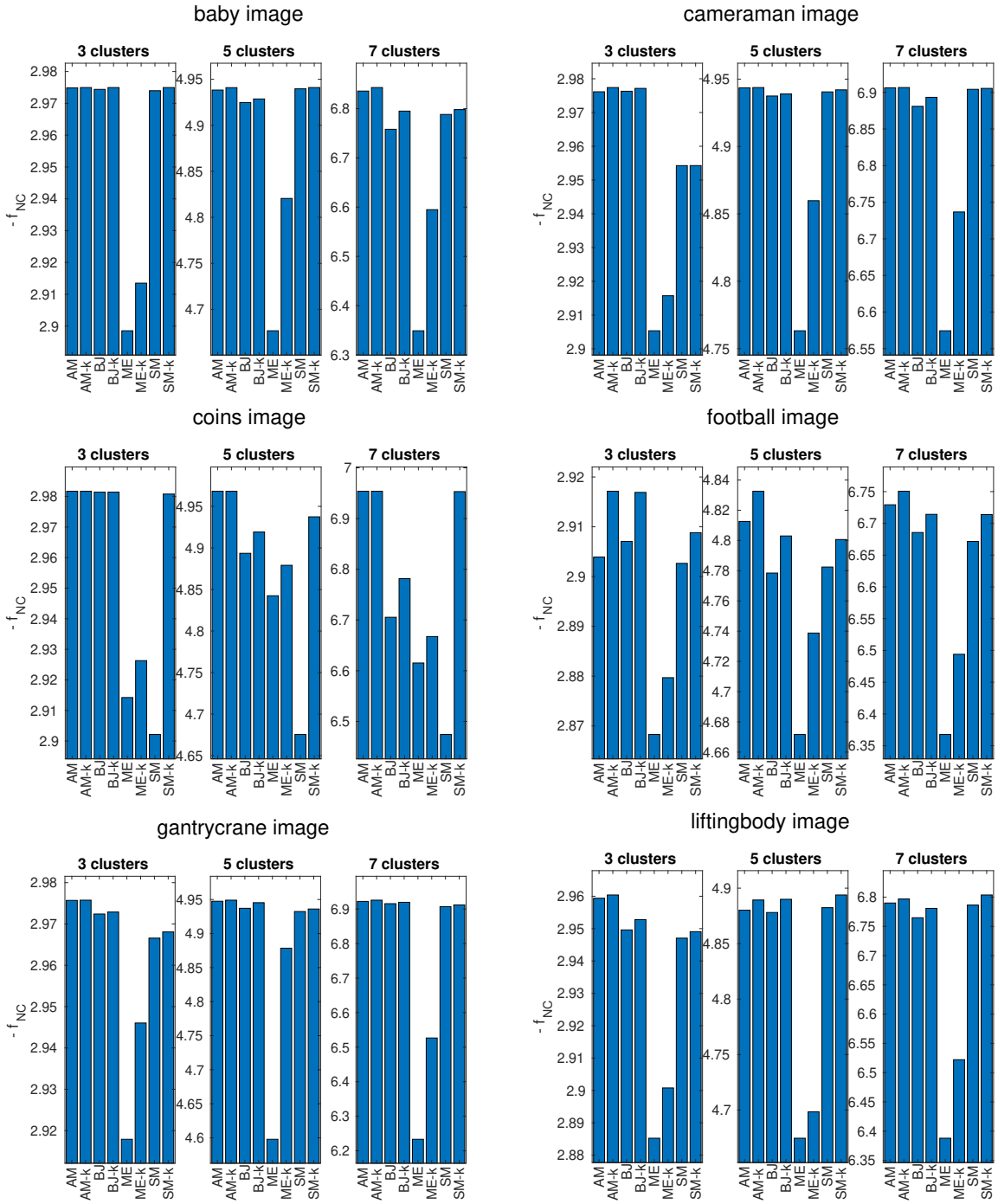


Figure 6.5: An average of 10 random runs on baby, cameraman, coins, football, gantrycrane, and liftingbody is reported. y-axis represents the function values. Multiple numbers of clusters are tested.

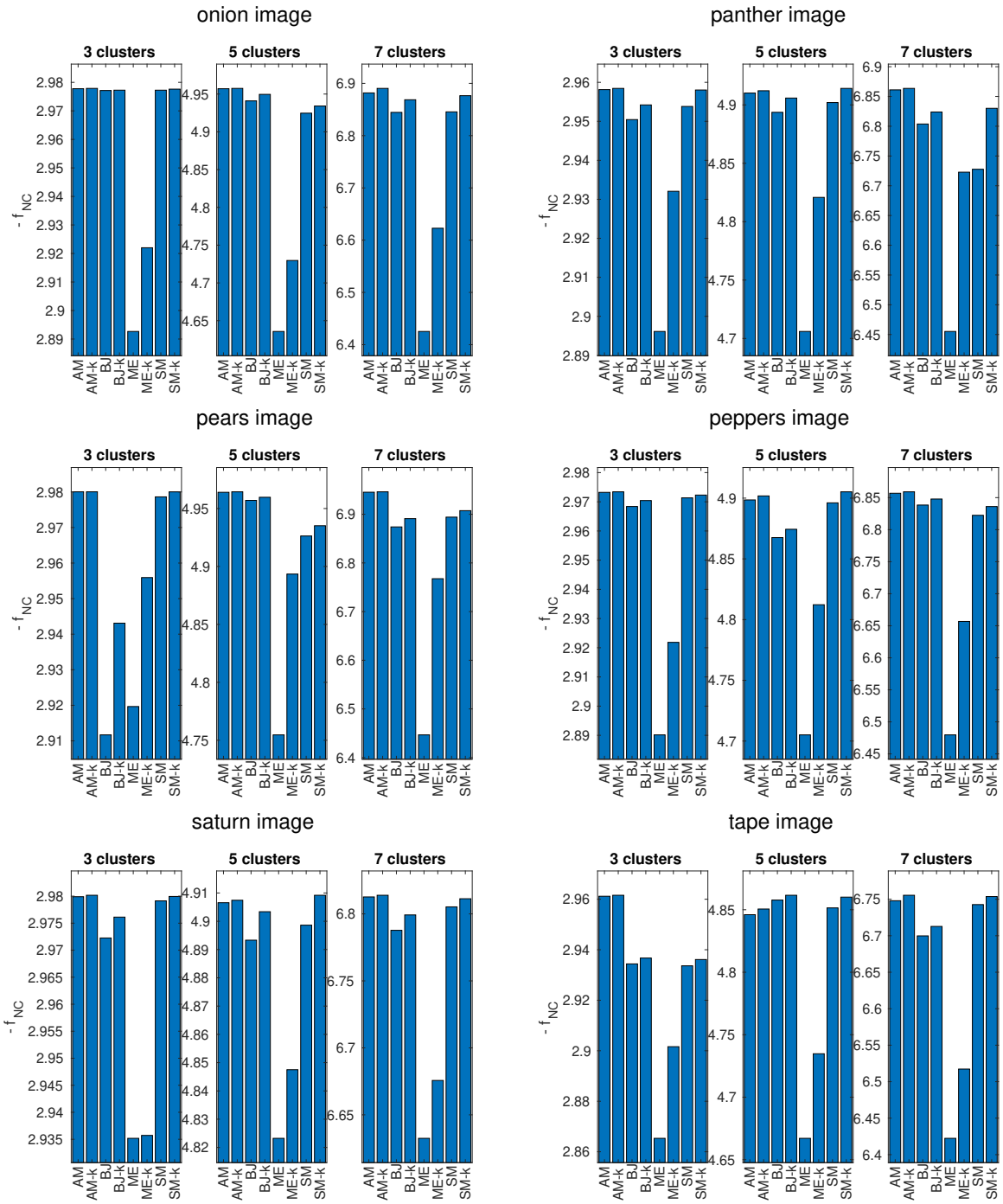


Figure 6.6: An average of 10 random runs on onion, panther, pears, peppers, saturn, and tape is reported. y-axis represents the function values. Multiple numbers of clusters are tested.

CHAPTER 7

RECURSIVE INEXACT ACCELERATED MANIFOLD PROXIMAL GRADIENT OPTIMIZATION ALGORITHM

For the I-AManPG algorithm, we fix the number of communities q ahead of time and set q as one input of I-AManPG. But in reality, the number of communities q is unknown, so we may improve the algorithm by adapting the parameter q. Besides that, to improve the time and space efficiency of I-AManPG, it is necessary to design some recursive version for I-AManPG. In this chapter, we propose a recursive version for the I-AManPG algorithm in Algorithm 18, and compare the performance of the recursive I-AManPG with the base I-AManPG on the community detection problem.

7.1 The Proposed Recursive Version of I-AManPG Algorithm

To design the recursive I-AManPG Algorithm 18, we use the similar logic of the Newman's spectral method [82] except that we change the bisection from spectral partitioning according to the signs of the leading eigenvector of the modularity matrix to the partitioning by I-AManPG with $q = 2$. The standard approach to find more than two clusters is repeated division into two: we use the I-AManPG with $q = 2$ first to divide the network into two partitions, then divide those parts, and so forth. In the community detection problem, it is not correct, after first dividing a network into two, to simply delete the edges falling between the parts and then apply the algorithm again to each subgraph [82]. This is because the degrees in the modularity matrix $M_{ij} = A_{ij} - \frac{k_i k_j}{2m}$ will change if edges are deleted, and any subsequent optimization of cost function would thus optimize the wrong quantity. Instead, the correct approach is to define a new modularity matrix $M^{(g)}$, where $M^{(g)}_{ij} = A_{ij} - \frac{k_i k_j}{2m} - \delta_{ij}[k^{(g)}_i - k_i \frac{d_g}{2m}]$, where $k^{(g)}_i$ is the degree of vertex i in subgraph g and d_g is the sum of the total degrees k_i of the vertices in the subgraph. So the subgraph modularity is $Q_g = \mathrm{tr}(X^T M^{(g)} X)$.

Algorithm 18 Recursive I-AManPG Algorithm

queue{1}=[1:n]
$\Delta Q = \infty$
while $length(queue) > 0$ **do**
    G=queue{1};
    $\hat{B}_G = B_G - diag(B_G * ones(length(G), 1))$
    $[V, E] = eigs(\hat{B}_G, length(G), eye(length(G)), 1, 'largestreal', 'IsFunctionSymmetric', 1);$
    if $(max(diag(E))) <= 10^{-5}$ **then**
        queue=queue(2:length(queue));
        modules=[modules G];
        continue
    end if
    Do bisection by calling I-AManPG algorithm with modularity matrix as $\hat{B_G}$ and $q = 2$
    comm1=one of the bisected groups and comm2 = the other one
    $\Delta Q = \mathrm{tr}(\hat{X}^T \hat{B_G} \hat{X})$
    if $\Delta Q < 10^{-5}$ **then**
        queue=queue(2:length(queue));
        modules=[modules G];
        continue
    end if
    [~, p]=size(modules);
    queue=[queue G(comm1) G(comm2)];
    queue=queue(2:length(queue));
end while
return modules

If there is no division of a subgroup that will increase the modularity of the network, then there is nothing to be gained by dividing the subgraph and then it should be left along (indivisible). This happens when here are no positive eigenvalues to the matrix $M^{(g)}$, and thus the leading eigenvalues provides a simple check for the termination of the subdivision process. The absence of positive eigenvalues is a sufficient but not necessary condition for indivisibility. We then simply calculate the modularity contribution for each proposed split directly and confirm that it is greater than 0.

Thus the recursive algorithm is summarized as follows. We construct the modularity matrix for the network and find its leading eigenvalue. We divide the network into two parts by I-AManPG with $q = 2$. If at any stage we find the proposed split makes a zero or negative contribution to the total modularity, we leave the corresponding subgraph undivided. When the entire network has been divided into indivisible subgraphs in this way, the algorithm ends. Or if we hit a desired number, then the algorithm ends.

7.2 Comparison with Other Algorithms on Applications

7.2.1 Comparison with Other Algorithms on LFR Benchmarks

In this section, we compare the recursive I-AManPG algorithm with other algorithms on detecting community structures of LFR benchmark networks [65].

Each result in Table 7.1 is an average result of 10 runs over 10 random runs for the graphs generated randomly as LFR benchmark networks. q_c is the computed number of communities,"force_q" algorithms denote the algorithms by adding a forcing criteria such that the algorithms stop when q_c is smaller than or equal to $q_{true} = 20$. From Table 7.1, we can find that the recursive I-AManPG algorithm is more efficient than the other methods in terms of the computation time. Regarding the effectiveness of the recursive I-AManPG algorithm, we can analyze it by comparing to the base I-AManPG version first and then comparing to Danon, Louvain and Newman's algorithms. When we compare the recursive I-AManPG algorithm to the base I-AManPG version, we can get the following results. For the cases of $\mu_{LFR} = 0, 0.1, 0.2$, even though the modularity are slightly smaller than the base I-AManPG version, the purity of the partition results by the recursive I-AManPG are all 1. This means that the partitions are very close to the ground-truth partition, furthermore the ground-truth partition is embedded into the partitions for some cases. For the other cases of μ_{LFR}, the results of the recursive I-AManPG algorithm are also close to the results of I-AManPG.

Table 7.1: Compare the effectiveness of Recursive I-AManPG to other state-of-the-art methods.

		μ_{LFR}								
		0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
Danon	NMI	0.9998	0.9891	0.9394	0.8504	0.7331	0.5808	0.3781	0.1412	0.0548
	AMI	0.9998	0.9870	0.9166	0.7922	0.6399	0.4736	0.2878	0.0935	0.0215
	Mod.	0.9496	0.8436	0.7225	0.5920	0.4687	0.3452	0.2458	0.1892	0.1814
	purity	0.9999	0.9938	0.9739	0.9414	0.9058	0.8344	0.6886	0.4269	0.3095
	time	2.8304	2.8597	2.8199	2.7648	2.7287	2.8318	2.7739	2.7625	2.7318
	q_c	20	20	18	15	11	9	7	7	8
Danon_force_q	NMI	0.9998	0.9889	0.9397	0.8494	0.7309	0.5794	0.3939	0.1778	0.0954
	AMI	0.9998	0.9870	0.9201	0.7962	0.6415	0.4727	0.2950	0.1035	0.0282
	Mod.	0.9496	0.8433	0.7209	0.5868	0.4615	0.3394	0.2431	0.1872	0.1794
	purity	0.9999	0.9935	0.9714	0.9349	0.8954	0.8198	0.6778	0.4201	0.3065
	time	2.8353	2.9606	2.8222	2.8068	2.8055	2.8959	2.8183	2.8459	2.7432
	q_c	20	20	20	20	20	20	20	20	20
Louvain	NMI	1.0000	1.0000	1.0000	1.0000	1.0000	0.9987	0.9805	0.2862	0.0784
	AMI	1.0000	1.0000	1.0000	1.0000	1.0000	0.9974	0.9652	0.2249	0.0358
	Mod.	0.9497	0.8499	0.7503	0.6500	0.5499	0.4496	0.3477	0.2098	0.1967
	purity	1.0000	1.0000	1.0000	1.0000	1.0000	0.9999	0.9950	0.4734	0.2660
	time	0.5444	0.7291	1.3703	1.8963	2.6797	3.3418	4.5768	9.2669	8.8130
	q_c	20	20	20	20	20	20	19	11	11
Louvain_force_q	NMI	1.0000	1.0000	1.0000	1.0000	1.0000	0.9987	0.9805	0.2981	0.0847
	AMI	1.0000	1.0000	1.0000	1.0000	1.0000	0.9974	0.9652	0.2382	0.0392
	Mod.	0.9497	0.8499	0.7503	0.6500	0.5499	0.4496	0.3477	0.2098	0.1967
	purity	1.0000	1.0000	1.0000	1.0000	1.0000	0.9999	0.9950	0.4642	0.2418
	time	0.5440	0.7478	1.0333	1.2042	1.7002	2.0761	2.7676	5.4529	5.5061
	q_c	20	20	20	20	20	20	19	12	12
Newman_Eig sparse function	NMI	0.9988	0.7225	0.7132	0.6760	0.5498	0.3912	0.2807	0.1340	0.0497
	AMI	0.9985	0.6521	0.6396	0.6122	0.4704	0.3071	0.2098	0.0907	0.0235
	Mod.	0.9482	0.5493	0.5057	0.4157	0.3051	0.2379	0.1917	0.1578	0.1461
	purity	0.9994	0.7828	0.8134	0.7460	0.6723	0.6366	0.5654	0.4367	0.3594
	time	0.6346	0.4391	0.4197	0.4522	0.4140	0.3333	0.3565	0.3130	0.3026
	q_c	20	24	21	19	15	9	7	6	6
Newman_Eig_force_q sparse function	NMI	0.9988	0.6831	0.6787	0.6674	0.5498	0.3912	0.2807	0.1340	0.0497
	AMI	0.9985	0.5996	0.5991	0.6026	0.4704	0.3071	0.2098	0.0907	0.0235
	Mod.	0.9482	0.4747	0.4463	0.4004	0.3051	0.2379	0.1917	0.1578	0.1461
	purity	0.9994	0.7998	0.8001	0.7358	0.6723	0.6366	0.5654	0.4367	0.3594
	time	0.6458	0.4666	0.4373	0.4528	0.4231	0.3418	0.3656	0.3213	0.3114
	q_c	20	18	17	18	15	9	7	6	6
I-AManPG	NMI	1.0000	1.0000	1.0000	1.0000	1.0000	0.9998	0.9600	0.4517	0.1294
	AMI	1.0000	1.0000	1.0000	1.0000	1.0000	0.9998	0.9539	0.4037	0.0563
	Mod.	0.9497	0.8499	0.7503	0.6500	0.5499	0.4498	0.3416	0.1735	0.1113
	purity	1.0000	1.0000	1.0000	1.0000	1.0000	0.9999	0.9679	0.5605	0.3044
	time	0.6357	0.4693	0.5870	0.9494	0.6749	0.4720	1.0332	1.6307	1.6757
	q	20	20	20	20	20	20	20	20	20
Recursive I-AManPG	NMI	0.9883	0.9846	0.9883	0.9885	0.9861	0.9278	0.7002	0.2280	0.1125
	AMI	0.9753	0.9679	0.9753	0.9767	0.9839	0.9219	0.6670	0.1657	0.0436
	Mod.	0.9454	0.8440	0.7469	0.6468	0.5426	0.4159	0.2666	0.1399	0.1204
	purity	1.0000	1.0000	1.0000	0.9994	0.9921	0.9538	0.7909	0.4289	0.2389
	time	0.0134	0.0194	0.0181	0.1529	0.1213	0.2216	0.3762	0.2576	0.1662
	q_c	19	19	19	19	20	20	19	15	20

Comparing to the base I-AManPG version, another important advantage is that we do not need to set the number of communities as one input in the recursive I-AManPG and we can get the well nested partitions with the ground-truth partition as expected. Now, let us compare the results of the recursive I-AManPG algorithm with Danon, Louvain and Newman's algorithms. Overall, Louvain algorithm can get the best quality of partitions over all the LFR benchmarks among these 4 algorithms, and the recursive I-AManPG can get the second best quality of partitions but with the fastest computation.Danon algorithm works well for the cases of $\mu_{LFR} = 0, 0.1, 0.2, 0.3$ and then degrades. Newman's algorithm degrades quickly for the noisy cases of LFR networks.

In addition, the NMI, AMI of the recursive I-AManPG degrade somewhat from Louvain and the base I-AManPG, but the purity indicates that the recursive I-AManPG gets still grouped networks correctly. Future work would be required on termination criteria. Note that Louvain has the same thing happen to it when we look at $\mu_{LFR} = 0.6$. We know that the base I-AManPG performs very close to Louvain and so the recursive I-AManPG is a very efficient time and space approach clearly competitive with the best algorithm, that is, the Louvain algorithm.

7.2.2 Comparison with Other Algorithms on Real World networks

Here we consider the four widely used real-world networks described in Section 6.1.1 for performance evaluation on the I-AManPG algorithm.

From Table 7.2, it can be seen that overall the recursive I-AManPG is more efficient than the base I-AManPG algorithm in terms of the computation time. As the network becomes larger, the recursive I-AManPG algorithm clearly demonstrates its superior efficiency. We can see this from the computation time on the network of Email. Generally speaking, the effectiveness of recursive I-AManPG is better than Danon and Newman's algorithm and close but not quite as good as I-AManPG and Louvain's algorithm in terms of NMI, AMI and purity.

If we look into the Karate club network more closely, we can find the purity of the recursive I-AManPG algorithm on Karate Club is 1. The computed communities of recursive I-AManPG to Karate Club are communities {1, 2, 3, 4, 5, 6, 7, 8, 11, 12, 13, 14, 17, 18, 20, 22}, {24, 35, 26, 28,32}, and {9, 10, 15, 16, 19, 21, 23, 27, 29, 30, 31, 33, 34}. They are the subgroups of the ground-truth groups. The ground-truth partition has two groups. One of group of ground-truth groups is the same as community 1 and the other group is the union of community 2 and 3 of the

Table 7.2: Compare the effectiveness of I-AManPG to other state-of-the-art methods. Each result is an average result of 10 random runs for the real-world networks (the best performance is in bold.)

		Football	Karate-2	Polbooks	Email
Danon	NMI	0.7298	0.5305	0.5740	0.5261
	AMI	0.5744	0.3830	0.5075	0.3715
	Mod.	0.5661	0.4087	**0.5225**	**0.4240**
	purity	0.9043	0.9412	**0.8571**	0.8587
	time	0.0237	0.0028	0.0233	2.8354
	q_c	6	4	4	26
Danon_force_q	NMI	0.7031	0.5305	0.5740	0.5531
	AMI	0.5490	0.3830	0.5075	0.3954
	Mod.	0.5183	0.4087	**0.5225**	0.4188
	purity	0.5826	0.9412	0.8571	0.4119
	time	**0.0232**	**0.0026**	**0.0228**	2.7915
	q_c	12	4	4	42
Louvain	NMI	0.8903	0.5866	**0.5745**	0.5296
	AMI	0.8208	0.4254	**0.5560**	0.3763
	Mod.	**0.6046**	**0.4188**	0.4986	0.4100
	purity	**0.9217**	0.9706	0.8476	**0.8736**
	time	0.0587	0.0251	0.0604	1.7888
	q_c	10	4	3	25
Louvain_force_q	NMI	0.8903	0.5866	**0.5745**	0.5296
	AMI	0.8208	0.4254	**0.5560**	0.3763
	Mod.	**0.6046**	0.4188	0.4986	0.4100
	purity	**0.9217**	0.9706	0.8476	**0.8736**
	time	0.0448	0.0191	0.0448	2.0603
	q_c	10	4	3	25
Newman_Eig Sparse	NMI	0.6987	0.6771	0.5201	0.5176
	AMI	0.5611	0.4936	0.4429	0.3805
	Mod.	0.4926	0.3934	0.4672	0.3951
	purity	0.8087	**1.0000**	0.8476	0.4100
	time	0.0311	0.0082	0.0312	0.3794
	q_c	8	4	4	43
Newman_Eig_force_q Sparse	NMI	0.6987	0.8041	0.5201	0.5176
	AMI	0.5611	0.6618	0.4429	0.3805
	Mod.	0.4926	0.2064	0.4228	0.3950
	purity	0.8087	**1.0000**	0.8476	0.4100
	time	0.0314	0.0065	0.0291	0.3804
	q_c	8	2	3	42
I-AManPG	NMI	**0.9047**	**1.0000**	0.4874	**0.5908**
	AMI	**0.8652**	**1.0000**	0.4756	**0.4645**
	Mod.	0.5875	0.3715	0.4878	0.2419
	purity	0.9130	**1.0000**	0.8190	0.5692
	time	0.2214	1.5514	0.0582	13.4378
	q	12	2	3	42
Recursive I-AManPG	NMI	0.8331	0.8155	0.4732	0.3531
	AMI	0.7328	0.6781	0.4575	0.2181
	Mod.	0.5965	0.3835	0.4376	0.2921
	purity	0.9043	**1.0000**	0.8286	0.7114
	time	0.0263	0.7580	0.9022	**0.2866**
	q	9	3	3	10

recursive I-AManPG algorithm. This means we can get well nested partitions with the ground-truth partition.

These empirical results support the view that the recursive I-AManPG algorithm improves the efficiency of the basic I-AManPG algorithm while not suffering significant loss in the quality of the solution. Even when the algorithm is stopped with fewer than the number of clusters in the ground-truth the high purity values indicate it produces suitably nested clustering. To further improve the efficiency of the recursive I-AManPG algorithm implementation, the main result of the recursive I-AManPG that will go in ROPTLIB [52] along with the non recursive I-AManPG.

CHAPTER 8

CONCLUSIONS AND FUTURE WORK

In this dissertation, two Riemannian optimization approaches for the clustering problems in terms of assignment matrix are proposed, which are the ARPPG method, and the I-AManPG method. To derive the I-AManPG method, the manifold structure of the feasible set is constructed efficiently. The global convergence analysis for the I-AManPG method is shown. To evaluate the performance of our approaches, we design numerical experiments testing the performance of the different algorithms on the community detection problems, the k-means model, the discriminative k-means model, and the normalized cut problem in the dissertation. In addition, the recursive I-AManPG algorithm is proposed to improve the time and space complexity of the solution process using I-AManPG. We have achieved the goal of the thesis statement of this dissertation. We have systematically explored the careful use of geometry, cost formulation, algorithmic rigor and efficient computational design to produce an approach flexible enough to handle graph-based and Euclidean data-based optimization problems arising in a range of well-known clustering related problems. The result is a prime candidate for further exploration of the computational implementation and related application problems.

8.1 Completed Work

The major contributions of this dissertation are:

1. **The constrained nonsmooth Riemannian optimization problem over a Stiefel manifold is solved by modifying the accelerated Riemannian manifold proximal gradient method to the accelerated Riemannian manifold projected proximal gradient method (ARPPG) by adding the projection (3.4).**

   We formulated the community detection problem as a constrained nonsmooth optimization problem on the compact Stiefel manifold. A Riemannian projected proximal gradient method is proposed and used to solve the problem. We applied the Riemannian optimization approach to the community detection problem. Numerical experimental results on synthetic LFR benchmarks and real-world networks show that ARPPG is effective and outperforms several state-of-art algorithms.

2. **The feasible set $\mathcal{F}_v = \{X \in \mathbb{R}^{n\times q} : X^TX = I_q, v \in \text{span}(X)\}$, with $v > 0$ is proved to be an embedded submanifold of $\text{St}(q, n)$. The manifold structure of the feasible set $\mathcal{F}_v$ is constructed.**

   We showed that the feasible set $\mathcal{F}_v$ is a Riemannian submanifold, and particularly an embedded submanifold of $\text{St}(q, n)$. The manifold structure of $\mathcal{F}_v$ is constructed, and it includes the tangent space, the normal space, the efficient retraction. Then we applied the accelerated Riemannian manifold proximal gradient method (AManPG) in [53] over the feasible set $\mathcal{F}_v$ to the clustering problems.

3. **Solving approximately subsproblem of AManPG by semi-smooth Newton method (I-AManPG) over the feasible set $\mathcal{F}_v$ is proposed.**

   To speed up the AManPG algorithm over the feasible set $\mathcal{F}_v$, we apply the semi-smooth Newton method [119], [69] to solve (3.6) approximately. The general idea of semi-smooth Newton method is to solve a system of nonlinear equations based on the generalized Jacobian. So, we need to reduce the optimization problem to a system of nonlinear equations in order to use the semi-smooth Newton method. This can be obtained by considering the KKT conditions. The corresponding algorithm is shown in Algorithm 13. In addition, the global convergence analysis for Algorithm 13 is shown in this dissertation.

4. **We apply AManPG and I-AManPG to the community detection problem by formulating the community detection problem as a nonsmooth optimization problem on the feasible set manifold $\mathcal{F}_v$.**

   We compared AManPG with I-AManPG on the LFR benchmark networks. The results show that I-AManPG can find the same partitioning as AManPG and I-AManPG is more efficient than AManPG in terms of computational time. We also compared I-AManPG with 3 other state-of-art algorithms, namely, Danon et.al's algorithm, Louvain Algorithm and Newman's spectral algorithm on the LFR benchmark networks and 4 real-world networks. Numerical performance comparisons show that the I-AManPG algorithm is effective, robust and competitive with existing algorithms.

5. **Besides the community detection problem, we also apply I-AManPG algorithm to the k-means model, the discriminative k-means model, and the normalized cut problem.**

   The k-means model can be rewritten as an alternating minimization algorithm for solving the optimization problem

$$\min_{X\in\mathcal{A}_{n,q}} ||A - XX^TA||_F^2, \tag{8.1}$$

where A is a data matrix where each data point in $\mathbb{R}^d$ corresponds to a row of A. I-AManPG over the feasible set is applied to k-means model and the results are comparable with the standard algorithms.

The discriminative k-means model is fit for the data sets live in a very high dimensional space. The discriminative k-means model is given by

$$\min_{X \in \mathcal{A}_{n\times q}, \lambda_{regu}>0} \operatorname{trace}(X^T(I_n + \frac{1}{\lambda_{regu}} A^T A)^{-1} X) + \log\det(I_n + \frac{1}{\lambda_{regu}} A^T A), \tag{8.2}$$

where $\lambda_{regu} > 0$ is the regularization parameter. This problem is solved by alternating between the computation of X for a given λ_{regu} and the computation of λ_{regu} for a given X. The former one is in the form of (1.5), therefore I-AManPG can be applied to this model.

The normalized cut method has been widely used for image segmentation. The initialization plays an important role in finding an acceptable solution to the normalized cut problem. We applied an initialization method based on I-AManPG to the normalized cut problem and this method is more effective than the state-of-the-art methods.

6. **We use a continuation technique for the balancing parameter of λ to improve the performance of our algorithms.**

   To improve the performance of ARPPG and I-AManPG, we proposed and evaluated a continuation technique to adapt the choice of the balancing parameter λ. This technique can improve the effectiveness of ARPPG and I-AManPG to some extent.

7. **Recursive I-AManPG Algorithm:**

   When the number of communities q is not reliably known or is too large, the I-AManPG algorithm can have difficulties with time and space complexity. Therefore, we proposed and evaluated a recursive version of I-AManPG algorithm by adapting the parameter q to reduce storage and increase the speed of the algorithm especially for large data sets.

8.2 Future Work

1. The main results of the I-AManPG algorithm and the recursive I-AManPG algorithm will go in the C++ library ROPTLIB [52], which will improve the efficiency of the algorithms.

2. We will conduct more applications to evaluate the performance of our algorithms.

   (a) The orthogonal nonnegative matrix factorization (ONMF) problem can be formulated as follows. Given an m-by-n nonnegative matrix M and a factorization rank k (with $k < n$), solve the following problem

   $$\min_{U\in\mathbb{R}^{m\times k}, V\in\mathbb{R}^{k\times n}} ||M - UV||_F^2, \text{ s.t. } U \geq 0,\ V \geq 0, VV^T = I_k. \tag{8.3}$$

In [90], Pompili et.al. show the equivalence of ONMF with a weighted variant of spherical k-means.The ONMF problem relates to data clustering and the I-AManPG algorithm applied to the ONMF problem as an additional demonstration of the efficacy and efficiency of Riemannian algorithms in this problem area.

(b) Community Detection: There are other cost functions for community detection that can be more effective than simple modularity.

For example, we can write for the RB model [92]

$$\begin{aligned} M_{RB} &= A - \rho \frac{A\mathbf{1}_n\mathbf{1}_n^T A}{\mathbf{1}_n^T A\mathbf{1}_n} \\ &= A - \frac{A\mathbf{1}_n\mathbf{1}_n^T A}{\mathbf{1}_n^T A\mathbf{1}_n} + \frac{A\mathbf{1}_n\mathbf{1}_n^T A}{\mathbf{1}_n^T A\mathbf{1}_n} - \rho\frac{A\mathbf{1}_n\mathbf{1}_n^T A}{\mathbf{1}_n^T A\mathbf{1}_n} \\ &= M_N + (1-\rho)\frac{A\mathbf{1}_n\mathbf{1}_n^T A}{\mathbf{1}_n^T A\mathbf{1}}, \end{aligned}$$

where M_N has rank $q-1$ and M_{RB} has rank q (ideal case).

For $A = ZZ^T$, we saw that we could write the "good" assignment matrix as $[X_* \mid \frac{\mathbf{1}}{\sqrt{n}}] \in St(q,n)$ and that X_* had to maximize $tr(X_*^T M_N X_*)$ while $X_* \in St(q-1,n)$ and $X_* \perp \frac{\mathbf{1}_n}{\sqrt{n}}$.

This is still true for M_{RB}, since $\frac{\mathbf{1}_n}{\sqrt{n}}$ is in the span of $Z\mathbf{1}_q = \mathbf{1}_n$, and hence for $X_* \perp \mathbf{1}_n$, we have

$$\max_{X_* \in St(q-1,n),\ X_* \perp \mathbf{1}_n} tr(X_*^T M_{RB} X_*) = \max tr(X_*^T M_N X_*) \tag{8.4}$$

To show (8.4),

$$\begin{aligned} LHS\ of\ (8.4) &= tr(X_*^T M_N X_* + (1-\rho)\frac{X_*^T A\mathbf{1}_n\mathbf{1}_n^T A X_*}{\mathbf{1}_n^T A\mathbf{1}}) \\ &= tr(X_*^T M_N X_*) + (1-\rho)\frac{\mathbf{1}_n^T A X_* X_*^T A\mathbf{1}_n}{\mathbf{1}_n A\mathbf{1}_n} \end{aligned}$$

Experiments with the RB-CNM model with various resolution parameters could yield significant improvement in the real-world networks where we have seen simple modularity have problems.

(c) Bipartite Networks: Bipartite network analysis is an important case of community detection problem. When modeling relations between two different classes of objects, bipartite graphs very often arise naturally. For instance, a graph of football players and clubs, with an edge between a player and a club if the player has played for that club, is a natural example of an affiliation network, a type of bipartite graph used in social network analysis. There are many other useful applications on bipartite network analysis, such as the railway optimization problem and modern coding theory etc. Applying our algorithms to the bipartite networks is very meaningful.

(d) Role Extraction Problem: The role models which subsume community detection including bipartite and others use a cost function like Reichardt and White quality function in [95].

Reichardt and White quality function as follows:

$$\mathcal{Q}_{RW}(\sigma, B) = \frac{1}{m} \sum_{i,j \in V} [a_{ij} A_{ij} B(\sigma_i, \sigma_j) + b_{ij}(1 - A_{ij})(1 - B(\sigma_i, \sigma_j))], \tag{8.5}$$

where $\sigma_i \in \{1, ..., q\}$ is the role to node i, and $B(\sigma_i, \sigma_j) = 1$ if links going form nodes of type σ_i to nodes of type σ_j, and a_{ij} and b_{ij} are some contributions. This cost function is a straightforward use of a generalized modularity type cost function. It is worthwhile to generalize I-AManPG to solve the role extraction problem.

3. We will investigate more effective and efficient recursive I-AManPG algorithms.

   The recursive I-AManPG algorithm in Chapter 7 is the initial version of the recursive algorithm to adapt the number of communities. The results are good, but it is worth to investigate much more effective and efficient recursive algorithms. Of particular interest are, the termination criterion, partitioning with a small number greater than 2 at each step perhaps adapted for each division, and the influence of the particular cost function used such as resolution limits.

4. Riemannain proximal gradient algorithms other than AManPG, I-AManPG, and recursive I-AManPG have been developed and continue as the subject of intense research. Of particular interest for our future work is the previously mentioned IRPG algorithm [54]. Its design is more concerned with its mathematical properties and less with computational efficiency. The lessons learned in this dissertation leading to the efficient recursive I-AManPG and any library versions developed subsequently will be combined with the theoretical insights of IRPG to create a more efficient and effective family of Riemannian proximal gradient methods for a wide range of problems.

APPENDIX A

BASIC CONCEPTS OF RIEMANNIAN OPTIMIZATION

To understand the Riemannian Optimization better, this section reviews some important concepts and definitions of Riemannian manifolds that are related to this dissertation. More details of these concepts and algorithms can be found in [48], [87], [66], [1], [7], [91] and [50].

A.1 The Problem of Optimization on Manifolds

The goal of the optimization on manifolds is to find an (global, or more reasonably local) optimum of a real-valued function f defined over a manifold, i.e.,

Problem A.1.1 (Optimization on Manifolds)**.**
Given: a manifold $\mathcal{M}$ and a smooth function $f : \mathcal{M} \to \mathbb{R}$.
Sought: an element x^* of $\mathcal{M}$ such that there is a neighborhood V of x^* in $\mathcal{M}$ with $f(x^*) \leq f(x)$ for all $x \in V$ ("minimizer") or such that there is a neighborhood V of x^* in $\mathcal{M}$ with $f(x^*) \geq f(x)$ for all $x \in V$ ("maximizer").

Roughly speaking, a manifold is a set endowed with a collection of coordinate patches that overlap smoothly, that is, a *manifold* is a set of points that is locally Euclidean. More precisely, each point of a d-dimensional manifold has a neighborhood that is homeomorphic to the Euclidean space of dimension d. A *smooth manifold (also differentiable manifold)* is a type of manifold that is locally similar enough to a linear space to allow one to do calculus. In formal terms, a smooth manifold is a topological manifold with a globally defined differential structure. Any topological manifold can be given a differential structure locally by using the homeomorphisms in its atlas and the standard differential structure on a linear space. A *Riemannian manifold or Riemannian space* $(\mathcal{M}, g)$ is a real, smooth manifold $\mathcal{M}$ equipped with a positive-definite inner product g_x on the tangent space $T_x\mathcal{M}$ at each point x.

An optimization problem generally requires taking derivatives and gradients of a function, so we need a smooth structure on a manifold $\mathcal{M}$ that allows us to do the calculation. A Riemannian

manifold is a good choice. So we consider the optimization on Riemannian manifolds in this dissertation.

Typically, Riemannian optimization is considered as unconstrained optimization on a constrained space and ideas from unconstrained optimization algorithms on a Euclidean space have been adapted to optimization on manifolds. However, to use these ideas, we must reconsider many basic definitions, constructs and algorithmic techniques, since extending them from the Euclidean space to the manifold is not trivial. For example, the addition and subtraction of two points in the Euclidean space is well-defined but does not extend, in general, to two points on a manifold.

Therefore, we review some ingredients of Riemannian manifolds and associated basic computations and some key Riemannian optimization algorithms in the following sections.

A.2 Tangent Space and Tangent Vector

In order to apply line search algorithms, we must consider the direction of motion on a manifold. Let $\gamma(t) : \mathbb{R} \to \mathcal{M} : t \mapsto \gamma(t)$ be a smooth mapping on $\mathcal{M}$ satisfying $\gamma(0) = x$, i.e., γ is a curve through x at $t = 0$. To define the direction at x along γ, let us consider a smooth real-valued function f on $\mathcal{M}$, then the function $f \circ \gamma : t \mapsto f(\gamma(t))$ is a smooth function from $\mathbb{R}$ to $\mathbb{R}$ with a well-defined classical derivative. If $\mathcal{F}_x(\mathcal{M})$ denotes the set of smooth functions defined on a neighborhood of x, then we can define $\dot{\gamma}(0)$ as a mapping from $\mathcal{F}_x(\mathcal{M})$ to $\mathbb{R}$

$$\begin{aligned} \dot{\gamma}(0) &= (f \circ \gamma)'(0) \\ &= \lim_{h \to 0} \frac{f(\gamma(h)) - f(\gamma(0))}{h}. \end{aligned} \tag{A.1}$$

This mapping is the direction at x along the curve γ and it is also called a tangent vector to the curve γ at $t = 0$. The formal definition of tangent vectors is as follows.

Definition A.2.1 (Tangent Vector)**.** *A tangent vector ξ_x to a manifold $\mathcal{M}$ at a point x is a mapping from $\mathcal{F}_x(\mathcal{M})$ to $\mathbb{R}$ such that there exists a curve γ on $\mathcal{M}$ with $\gamma(0) = x$, satisfying*

$$\xi_x f = \dot{\gamma}(0) f := \frac{d(f(\gamma(t)))}{dt}|_{t=0}, \tag{A.2}$$

for all $f \in \mathcal{F}(\mathcal{M})$. The curve γ is said to realize the tangent vector ξ_x. The point x is called the root of the tangent vector ξ_x.

The set of all tangent vectors at x is the *tangent space* to $\mathcal{M}$ at x, denoted by $\mathrm{T}_x\mathcal{M}$. The tangent space $\mathrm{T}_x\mathcal{M}$ is a vector space, i.e., closed under linear combination and has the same dimension as the manifold $\mathcal{M}$. So line searches are performed on the tangent space rather than on the manifold. Then we need to get back to the manifold by using an operation called *retraction* which will be discussed later.

The union of all tangent spaces is called the *tangent bundle* of the manifold, denoted by $\mathrm{T}\mathcal{M}$. A *vector field* is also an important concept, and it is a smooth mapping from $\mathcal{M}$ to the tangent bundle, $\xi : \mathcal{M} \to \mathrm{T}\mathcal{M} : x \mapsto \xi_x \in \mathrm{T}_x\mathcal{M}$. It assigns to each point a tangent vector. The set of all smooth vector fields on $\mathcal{M}$ is denoted by $\chi(\mathcal{M})$ and is endowed with the operations of addition of two vector fields and multiplication of a vector field by a function $f \in \mathcal{F}_x(\mathcal{M})$, i.e., for all $x \in \mathcal{M}$,

$$(\xi + \zeta)_x = \xi_x + \zeta_x,$$
$$(f\xi)_x = f(x)\xi_x.$$

A.3 Riemannian Metric

The tangent space at a point on the manifold provides us with a vector space that approximates the manifold locally. Endowing the tangent space with an inner product allows us to compute angles and lengths of tangent vectors.

A *Riemannian metric* g defined on the tangent spaces of x is a smoothly varying inner product $g_x : \mathrm{T}_x\mathcal{M} \times \mathrm{T}_x\mathcal{M} \to \mathbb{R}$, and denoted as

$$g_x(\xi, \eta) = < \xi, \eta >_x, \tag{A.3}$$

where $\xi, \eta \in \mathrm{T}_x\mathcal{M}$.

We define $\xi^\flat$ as a function from $\mathrm{T}_x\mathcal{M}$ to $\mathbb{R}$ such that $\xi^\flat\eta = g_x(\xi, \eta)$ for all $\eta \in \mathrm{T}_x\mathcal{M}$, where the notation, flat $\flat$, relates tangent vectors to the Riemannian metric. A *Riemannian manifold or Riemannian space* is the manifold $\mathcal{M}$ equipped with the inner product g_x on the tangent space $\mathrm{T}_x\mathcal{M}$ at each point x.

Then we can define a *distance metric* on $\mathcal{M}$ by using the norm induced by the inner product on the tangent space as follows:

$$d(x, y) = \inf_\gamma \{ \int_0^1 \sqrt{g_{\gamma(t)}(\dot\gamma(t), \dot\gamma(t))} dt \} = \inf_\gamma \{ \int_0^1 ||\dot\gamma(t)||_{g_{\gamma(t)}} dt \}, \tag{A.4}$$

where γ is a curve on $\mathcal{M}$ with $\gamma(0) = x$ and $\gamma(1) = y$.

Once we are equipped with the definition of distance on the manifold, then we can define *neighborhoods* on the manifold. The open ball of radius δ around x, denoted by $\mathcal{B}_\delta(x)$:

$$\mathcal{B}_\delta(x) = \{y \in \mathcal{M} : d(x, y) < \delta\}. \tag{A.5}$$

Therefore, we can define local minimizers for a function on a manifold. Given a function $f : \mathcal{M} \to \mathbb{R}$, a point x^* is a strict local minimizer if there exists some $\delta > 0$ such that

$$f(x^*) < f(y), \text{ for all } y \in \mathcal{B}_\delta(x^*).$$

A.4 Affine Connection, Geodesics, Exponential Mapping and Parallel Translation

Given a curve $\gamma(t)$ on a Riemannian manifold, $\dot{\gamma}(t)$ shows the direction along the curve, and the length of $\dot{\gamma}(t)$ shows the speed of change on the curve. To define the acceleration on manifolds, we need the concept of affine connection, which provides the idea of differentiating tangent vectors.

Definition A.4.1 (Affine Connection)**.** *Let $\mathcal{F}_x(\mathcal{M})$ denote the set of all smooth functions on a neighborhood of x, and $\chi(\mathcal{M})$ denote the set of smooth vector fields on $\mathcal{M}$. An affine connection ∇ on a manifold $\mathcal{M}$ is a mapping*

$$\nabla : \chi(\mathcal{M}) \times \chi(\mathcal{M}) \to \chi(\mathcal{M}) : (\xi, \eta) \mapsto \nabla_\xi \eta \tag{A.6}$$

that satisfies the following properties: for all $f, g \in \mathcal{F}_x\mathcal{M}, a, b \in \mathbb{R}$ and $\xi, \eta, \zeta \in \chi(\mathcal{M})$:

(i) $\mathcal{F}_x(\mathcal{M})$-linearity in the first argument: $\nabla_{f\eta+g\zeta}\xi = f\nabla_\eta\xi + g\nabla_\zeta\xi$;

(ii) $\mathbb{R}$-linearity in the second argument: $\nabla_\eta(a\xi + b\zeta) = a\nabla_\eta\xi + b\nabla_\eta\zeta$;

(iii) Product rule (Leibniz's Law): $\nabla_\eta(f\xi) = (\eta f)\xi + f\nabla_\eta\xi$.

The resulting vector field $\nabla_\eta\xi$ is called the covariant derivative of ξ with respect to η for the affine conncetion ∇.

There are infinitely many affine connections for any manifold $\mathcal{M}$. However, there are certain affine connections that may be preferred due to particular properties. On a Riemannian manifold $(\mathcal{M}, g)$, a preferred affine connection, called the Riemannian connection or *Levi-Civita connection*, satisfies the following two additional conditions:

(i) Symmetry: $(\nabla_\eta \xi - \nabla_\xi \eta) f = \eta(\xi f) - \xi(\eta f)$;

(ii) Compatibility with Riemannian metric: $\zeta g(\eta, \xi) = g(\nabla_\zeta \eta, \xi) + g(\eta, \nabla_\zeta \xi)$.

A curve γ on a Riemannian manifold $(\mathcal{M}, g)$ endowed with an affine connection ∇ is a *geodesic* if it has zero acceleration:

$$\nabla_{\dot\gamma(t)} \dot\gamma(t) := \frac{D^2}{dt^2}\gamma(t) := \frac{D}{dt}\dot\gamma(t) = 0, \tag{A.7}$$

for all t. With the Riemannian connection, one of the geodesics linking two points on the manifold is also a minimal length curve. In this dissertation, we only consider the Riemannian connection.

Given a point $x \in \mathcal{M}$ and a tangent vector $\eta \in \mathrm{T}_x\mathcal{M}$, there exists a unique geodesic $\gamma(t; x, \eta)$ satisfying $\gamma(0) = x$ and $\dot\gamma(0) = \eta$. In addition, the geodesic has the homogeneity property, $\gamma(t; x, a\eta) = \gamma(at; x, \eta)$. The mapping

$$\mathrm{Exp}_x : \mathrm{T}_x\mathcal{M} \to \mathcal{M} : \eta \mapsto \mathrm{Exp}_x \eta = \gamma(1; x, \eta), \tag{A.8}$$

is called the *exponential mapping* at x. A manifold $(\mathcal{M}, g)$ is called geodesically complete if and only if Exp_x is defined for all $x \in \mathcal{M}$ and all $\eta \in \mathrm{T}_x\mathcal{M}$. That is, every geodesic of a geodesically complete manifold can be extended indefinitely. When performing line search algorithms, exponential mapping allows us to move in the direction of a tangent vector in the tangent space, and then map the tangent vector to a point on the manifold.

We may need to compare or combine tangent vectors in different tangent spaces in many situations. Since the affine connection provides the idea of differentiating tangent vectors in different tangent spaces, we can use it to define the vector transport moving a tangent vector from one tangent space to another. In a Euclidean space the simplest such motion is parallel translation that is simply moving the root of the given vector to any other point in the space to yield a parallel vector field. For a Riemannian manifold parallel translation produces a suitably generalized notion of a parallel vector field along a single curve. A vector field ξ on a curve γ satisfying $\frac{D}{dt}\xi = \nabla_{\dot\gamma}\xi = 0$ is called *parallel*. Given $a \in \mathbb{R}$ in the domain of γ and $\xi_{\gamma(a)} \in \mathrm{T}_{\gamma(a)}\mathcal{M}$, there is a unique parallel vector field ξ on γ such that $\xi(a) = \xi_{\gamma(a)}$. The operator $P_\gamma^{b \leftarrow a}$ sending $\xi(a)$ to $\xi(b)$ is called *parallel translation* along γ. In other words, we have

$$\frac{D}{dt}(P_\gamma^{t \leftarrow a}\xi(a)) = 0. \tag{A.9}$$

If ∇ is the Riemannian connection, the resulting parallel translation is an isometry.

A.5 Riemannian Gradient and Riemannian Hessian

Gradient-based optimization requires the notion of a gradient as the direction of steepest ascent of an objective function. Second-order optimization algorithms, such as Newton's method, may require the Hessian. In the setting of manifolds, these concepts can be defined as follows.

Definition A.5.1 (Riemannian Gradient)**.** *Let f be a function defined on a Riemannian manifold $(\mathcal{M}, g)$. The Riemannian gradient of f at x, denoted as* $\mathrm{grad} f(x)$*, is the unique tangent vector in* $\mathrm{T}_x\mathcal{M}$ *satisfying*

$$< \mathrm{grad} f(x), \xi >_x = \mathrm{D} f(x)[\xi], \ \forall\ \xi \in \mathrm{T}_x\mathcal{M}, \tag{A.10}$$

where the directional derivative is denoted by $\mathrm{D}f$ *and the definition of a tangent vector identifies* $\mathrm{D}f(x)[\xi] = \xi f$*.*

Definition A.5.2 (Riemannian Hessian)**.** *Given a real-valued function f on a Riemannian manifold $(\mathcal{M}, g)$, the Riemannian Hessian of f at a point x in the direction of $\eta \in \mathrm{T}_x\mathcal{M}$, denoted by* $\mathrm{Hess} f(x)[\eta]$*, is the unique linear mapping*

$$\mathrm{Hess} f(x) : \mathrm{T}_x\mathcal{M} \to \mathrm{T}_x\mathcal{M}$$

that satisfies

$$\mathrm{Hess} f(x)[\eta] = \nabla_\eta \mathrm{grad} f(x), \tag{A.11}$$

for all $\eta \in \mathrm{T}_x\mathcal{M}$, where ∇ is the Riemannian connection chosen for $\mathcal{M}$.

From the symmetry of the Riemannian connection, we know the Hessian is a self-adjoint operator in terms of Riemannian metric, i.e.,

$$< \mathrm{Hess} f(x)[\eta], \xi >_x = < \eta, \mathrm{Hess} f(x)[\xi] >,$$

for all $\eta, \xi \in \mathrm{T}_x\mathcal{M}$.

A.6 Retraction and Vector Transport

Generally speaking, we perform line search or build a local model on the tangent space and find a tangent vector, and then we need a mapping to get the chosen tangent vector back to the manifold in order to obtain the next iterate. Such a mapping is called a retraction. A retraction

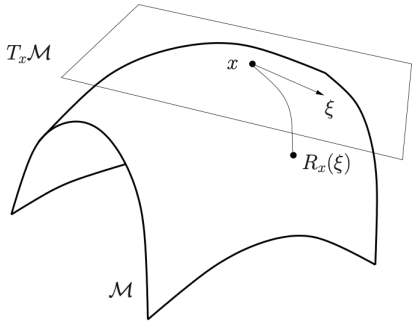


Figure A.1: Retraction

allows us to move in the direction of a tangent vector while staying on the manifold. We can refer to Figure A.1 for an illustration of the definition of a retraction.

The formal definition of a retraction is as follows.

Definition A.6.1 (Retraction)**.** *A retraction on a manifold $\mathcal{M}$ is a smooth mapping R from the tangent bundle $\mathrm{T}\mathcal{M}$ onto $\mathcal{M}$ with the following properties. Let R_x denote the restriction of R to $\mathrm{T}_x\mathcal{M}$.*

(i) $R(0_x) = x$ for all $x \in \mathcal{M}$, where 0_x denote the origin of $\mathrm{T}_x\mathcal{M}$.

(ii) With the canonical identification $\mathrm{T}_{0_x}\mathrm{T}_x\mathcal{M} \simeq \mathrm{T}_x\mathcal{M}$, R_x satisfies

$$\mathrm{D}R_x(0_x) = \mathrm{id}_{\mathrm{T}_x\mathcal{M}}, \tag{A.12}$$

where $\mathrm{id}_{\mathrm{T}_x\mathcal{M}}$ denotes the identity mapping on $\mathrm{T}_x\mathcal{M}$.

In Section A.4, we defined the exponential mapping which is a special retraction. When we use the exponential mapping, we actually move along the geodesic defined by the tangent vector. Besides mapping elements of $\mathrm{T}_x\mathcal{M}$ into points of $\mathcal{M}$, a retraction R_x can transform cost functions defined in a neighborhood of $x \in \mathcal{M}$ into cost functions defined on the vector space $\mathrm{T}_x\mathcal{M}$. Specifically, given a real-valued function f on a manifold $\mathcal{M}$ equipped with a retraction R, we let $\hat{f} = f \circ R$ denote the *pullback* of f through R. For $x \in \mathcal{M}$,

$$\hat{f}_x = f \circ R_x \tag{A.13}$$

is the restriction of $\hat{f}$ to $\mathrm{T}_x\mathcal{M}$.

Roughly speaking, a vector transport $\mathcal{T}$ on a manifold is a mapping that transport a tangent vector ξ from a point $x \in \mathcal{M}$ to a point $R_x(\eta) \in \mathcal{M}$, and it is built upon the retraction. We can refer to Figure A.2 as an illustration of the concept of a vector transport.

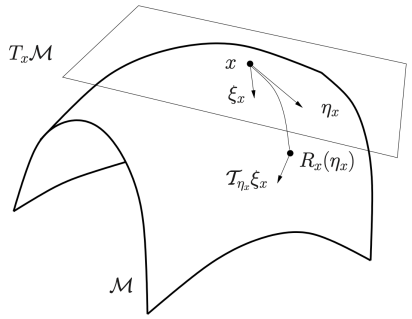


Figure A.2: Vector Transport

A vector transport is a more general notion closely related to the classical concept of parallel translation along geodesics. Like the exponential mapping, the parallel translation can be computationally demanding or cumbersome in numerical algorithms. Another vector transport provides an alternative to parallel translation and may reduce the the computational effort while retaining the convergence properties of the algorithm. A vector transport is defined formally as follows.

Definition A.6.2 (Vector Transport)**.** *A vector transport on a manifold $\mathcal{M}$ is a smooth mapping*

$$\mathcal{T} : \mathrm{T}\mathcal{M} \oplus \mathrm{T}\mathcal{M} \to \mathrm{T}\mathcal{M}, (\eta_x, \xi_x) \mapsto \mathcal{T}_{\eta_x}\xi_x$$

satisfying the following properties for all $x \in \mathcal{M}$, where the operation $\oplus$ is called the Whitney sum:

(i) (Associated Retraction) There exists a retraction R, called the retraction associated with $\mathcal{T}$, such that the following diagram in Figure A.3 commutes

$$\begin{array}{ccc} (\eta_x, \xi_x) & \xrightarrow{\mathcal{T}} & \mathcal{T}_{\eta_x}(\xi_x) \\ \downarrow & & \downarrow \pi \\ \eta_x & \xrightarrow[R]{} & \pi\left(\mathcal{T}_{\eta_x}(\xi_x)\right) \end{array}$$

Figure A.3: Associated Retraction Diagram

where $\pi(\mathcal{T}_{\eta_x}(\xi_x))$ denotes the foot of the tangent vector $\mathcal{T}_{\eta_x}(\xi_x)$.

(ii) (Consistency) $\mathcal{T}_{0_x}\xi_x = \xi_x$ *for all* $\xi_x \in \mathrm{T}_x\mathcal{M}$.

(iii) (Linearity) $\mathcal{T}_{\eta_x}(a\xi_x + b\zeta_x) = a\mathcal{T}_{\eta_x}(\xi_x) + b\mathcal{T}_{\eta_x}(\zeta_x)$.

Given a retraction R on a manifold $\mathcal{M}$, the vector transport based on differentiated retraction is an important approach to produce the vector transport, which is given by

$$\begin{aligned}\mathcal{T}_{\eta_x}(\xi_x) &= \mathrm{D}R_x(\eta_x)[\xi_x] \\ &= \frac{d}{dt}R_x(\eta_x + t\xi_x)\mid_{t=0}. \end{aligned} \tag{A.14}$$

The choice of retraction and vector transport is an important step in the design of efficient Riemannian optimization algorithms.

APPENDIX B

AN OVERVIEW OF PROJECTED PROXIMAL GRADIENT METHOD ON EUCLIDEAN SPACE

In this dissertation, we use projected gradient, proximal gradient method and the fast proximal gradient method (FISTA) on Riemannian space. They can be generated from Euclidean space. In this section, let us briefly review the projected gradient method, the projected subgradient method, the proximal gradient method, and the fast proximal gradient method (FISTA) in the Euclidean space. For more details, we refer the reader to [11] and [8]. We generalize the methods from Euclidean Space to Riemannian Space, see Chapter 3 and Chapter 4.

B.1 The Projected Gradient Method

Gradient descent is a standard way to solve unconstrained optimization problem. For constrained optimization problem, the projected gradient descent method is proposed by introducing a projection operator.

We mainly consider the following model

$$\min\{f(x) : x \in C\}, \tag{B.1}$$

where $f : \mathbb{E}$(i.e., Euclidean Space) $\to (-\infty, \infty]$ is proper closed, convex and continuously differentiable, $C \subseteq \mathbb{E}$ is nonempty closed and convex.

To solve this constrained optimization problem (B.1), we can generalize the gradient descent method to the following projected gradient descent method as shown in Algorithm 19.

Algorithm 19 Projected Gradient Method on Euclidean Space

Initialization: pick $x_0 \in C$ arbitrarily.

General Step: for any $k = 0, 1, 2, ...$ execute the following steps:

1: pick a stepsize $t_k > 0$ and the descent direction as $-\nabla f(x_k)$;
2: set $x_{k+1} = P_C(x_k - t_k \nabla f(x_k))$.

Each iteration of the projected gradient method consists of a step taken toward the negative of gradient and an orthogonal projection onto the underlying set C.

Remark B.1.1 (Reformulation of the update step 2 in Algorithm 19)**.** *To understand the role of the Euclidean norm in the definition of the projected gradient method, we consider the following reformulation of the update step 2:*

$$x_{k+1} = \underset{x \in C}{\operatorname{argmin}} \{f(x_k) + < \nabla f(x_k), x - x_k > + \frac{1}{2t_k}||x - x_k||^2\}. \tag{B.2}$$

We can get the equivalence between the two forms, Step 2 and (B.6)*, in the Euclidean case by using the following identity:*

$$f(x_k) + < \nabla f(x_k), x - x_k > + \frac{1}{2t_k}||x - x_k||^2 = \frac{1}{2t_k}||x - (x_k - t_k \nabla f(x_k))||^2 + D,$$

where D is a constant (D does not depend on x).

The reformulation form (B.2) *indicates that x_{k+1} is constructed by minimizing a linearization of the objective function plus a quadratic proximity term. This form is convenient to be used to generalize the projected gradient method to non-Euclidean setting.*

B.2 The Projected Subgradient Method

If f is not differentiable, then scheme in Algorithm 19 is not well defined. Under our convexity assumption, a natural generalization to the nonsmooth case will consist in replacing the gradient by a subgradient (assuming that it exists).

We mainly consider the following model

$$\min\{f(x) : x \in C\}, \tag{B.3}$$

where $f : \mathbb{E} \to (-\infty, \infty]$ is proper closed, convex and not differentiable, $C \subseteq \mathbb{E}$ is nonempty closed and convex, $C \subseteq int(dom(f))$, and the optimal set of (B.3) is nonempty and denoted by X^*, the optimal value of the problem is denoted by f_{opt}.

Definition B.2.1 (Subgradient)**.** *Let $f : \mathbb{E}$(i.e., Euclidean space$) \to (-\infty, \infty]$ be a proper function and let $x \in dom(f)$. A vector $g \in \mathbb{E}^*$(i.e., the dula space of Euclidean space) is called a subgradient of f at x if*

$$f(y) \geq f(x) + < g, y - x > \ \text{for all } y \in \mathbb{E}. \tag{B.4}$$

Definition B.2.2 (Subdifferential)**.** *The set of all subgradients of f at x is called the subdifferential of f at x and is denoted by $\partial f(x)$:*

$$\partial f(x) := \{g \in \mathbb{E}^* : f(y) \geq f(x) + < g, y - x > \text{ for all } y \in \mathbb{E}.\} \tag{B.5}$$

Definition B.2.3 (Subdifferentiability)**.** *A proper function $f : \mathbb{E} \to (-\infty, \infty]$ is called subdifferentiable at $x \in dom(f)$ if $\partial f(x) \neq \emptyset$.*

The collection of points of subdifferentiability is denoted by $dom(\partial f)$:

$$dom(\partial f) = \{x \in E : \partial f(x) \neq \emptyset\}.$$

Theorem B.2.4 (Nonemptiness and boundedness of the subdifferential set at interior points of the domain)**.** *Let $f : \mathbb{E} \to (-\infty, \infty]$ be a proper convex function, and assume that $\tilde{x} \in int(dom(f))$. then $\partial f(\tilde{x})$ is nonempty and bounded.*

Remark B.2.5 (Subdifferentiability of f and closedness of X^*)**.** *Since f is convex and $C \subseteq \int(dom(f))$ in problem* (B.3)*, it follows by Theorem B.2.4 that f is subdifferentiable over C. Also, since f is closed,*

$$X^* = C \cap Lev(f, f_{opt})$$

is closed. This means in particular that for any $x \notin X^*$ the distance $d_{X^*}(x)$ is positive.

Equipped with the above observations, we can speculate the projected subgradient method as follows in Algorithm 20. Each iteration of the projected subgradient method consists of a step taken

Algorithm 20 Projected Subgradient Method on Euclidean Space

Initialization: pick $x_0 \in C$ arbitrarily.

General Step: for any $k = 0, 1, 2, ...$ execute the following steps:

1: pick a stepsize $t_k > 0$ and a subgradient $f'(x_k) \in \partial f(x_k)$;
2: set $x_{k+1} = P_C(x_k - t_k f'(x_k))$.

toward the negative of the chosen subgradient and an orthogonal projection onto the underlying set C.

Remark B.2.6 (Reformulation of the update step 2 in Algorithm 20)**.** *Similarly as in projected gradient method, we can get the following reformulation of the update step 2:*

$$x_{k+1} = \operatorname*{argmin}_{x \in C} \{f(x_k) + < f'(x_k), x - x_k > + \frac{1}{2t_k} ||x - x_k||^2\}. \tag{B.6}$$

B.3 The Proximal Gradient Method

In this section, we focus on the composite model

$$\min_{x\in\mathbb{E}}\{F(x) := f(x) + g(x)\}, \tag{B.7}$$

where $g : \mathbb{E} \to (-\infty, \infty]$ is proper closed and convex, $f : \mathbb{E} \to (-\infty, \infty]$ is proper and closed, $dom(f)$ is convex, $dom(g) \subseteq int(dom(f))$, and f is L_f-smooth over $int(dom(f))$.

To understand the idea behind the method for solving (B.7), we start with the projected gradient method for solving (B.7) in the case where $g = \delta_C$ with C being a nonempty closed and convex set. In this case, the problem changes to

$$\min f(x) : x \in C. \tag{B.8}$$

The general update step of the projected gradient method for solving (B.8) is

$$x_{k+1} = P_C(x_k - t_k\nabla f(x_k)),$$

where t_k is the stepsize at iteration k. It is easy to be written as (see Remark B.1.1)

$$x_{k+1} = \operatorname*{argmin}_{x\in C}\{f(x_k) + < \nabla f(x_k), x - x_k > +\frac{1}{2t_k}||x - x_k||^2\}. \tag{B.9}$$

Back to the more general problem (B.7), it is natural to generalize the above idea and to define the next iteration as the minimizer of the sum of linearization of f around x_k, the nonsmooth function g, and a quadratic proximal term:

$$x_{k+1} = \operatorname*{argmin}_{x\in\mathbb{E}}\{f(x_k) + < \nabla f(x_k), x - x_k > +g(x) + \frac{1}{2t_k}||x - x_k||^2\}. \tag{B.10}$$

Once we do some simple algebraic manipulation and cancellation of constant terms, we can rewrite (B.10) as

$$x_{k+1} = \operatorname*{argmin}_{x\in\mathbb{E}}\{t_k g(x) + \frac{1}{2}||x - (x_k - t_k\nabla f(x_k))||^2\}. \tag{B.11}$$

By the definition of the proximal operator [89], (B.11) is the same as

$$x_{k+1} = prox_{t_k g}(x_k - t_k\nabla f(x_k)). \tag{B.12}$$

So the proximal gradient method can be described as in Algorithm 21.

Algorithm 21 Proximal Gradient Method on Euclidean Space

Initialization: pick $x_0 \in int(dom(f))$.

General Step: for any $k = 0, 1, 2, ...$ execute the following steps:

1: pick a stepsize $t_k > 0$;
2: set $x_{k+1} = prox_{t_k g}(x_k - t_k \nabla f(x_k))$.

B.4 The Accelerated Proximal Gradient Method (FISTA)

For the problem (B.7), the proximal gradient method achieves an $O(1/k)$ rate of convergence in function values to the optimal value under the assumptions that f is convex, Lipschitz-continuously differentiable, g is convex, and F is coercive [8]. Beck and Teboulle [9] devised an accelerated proximal gradient method, the fast iterative shrinkage-thresholding algorithm (FISTA) based on the Nesterov momentum technique. Under the same conditions as in the convergence analysis of the proximal gradient method, FISTA can obtain a rate of convergence of order $O(1/k^2)$.

FISTA can be described as in the following Algorithm 22.

Algorithm 22 FISTA on Euclidean Space

Input: (f, g, x_0), where f is convex, Lipschitz-continuously differentiable, g is convex, and $x_0 \in \mathbb{E}$.

Initialization: set $y_0 = x_0$ and $t_0 = 1$.

General Step: for any $k = 0, 1, 2, ...$ execute the following steps:

1: pick a stepsize $L_k > 0$;
2: set $x_{k+1} = prox_{\frac{1}{L_k} g}(y_k - \frac{1}{L_k} \nabla f(y_k))$;
3: set $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$;
4: Compute $y_{k+1} = x_{k+1} + (\frac{t_k - 1}{t_{k+1}})(x_{k+1} - x_k)$.

APPENDIX C

REVIEW OF BASIC ELEMENTS OF GRAPH THEORY

We consider network theory and applications in this dissertation. So we pay attention to some fundamentals of graph theory that are used throughout of this dissertation. Firstly, we define some graph notations and review some fundamental concepts used to characterize graph properties. Then, we describe some specific structural properties of graphs. In addition, we represent the notion of random graphs which are useful to model typical properties of networks or to quantify how similar a given network is from what one can observe on average in networks with similar properties. We refer the reader to [45], [115], [78], [19], [30], and [56] for additional resources on the graph theory.

C.1 Fundamental Concepts

A *graph* $G = (V, E)$ is a mathematical structure consisting of a finite set $V = \{1, 2, ..., n\}$ and a finite set $E = \{(i, j) \mid i, j \in V\}$. The elements of V are called *vertices (or nodes)*, while the elements of E are called *edges (or links)*. The cardinality of the set E i.e., the number of edges in the graph, is denoted by $\mid E \mid = m$. Each edge has a set of one or two vertices associated to it, which are called its *endpoints*. A vertex joined by an edge to a vertex i is said to be the *neighbors* of i. A pair (i, j) belongs to E, if there is connection between vertex i and vertex j. An edge (i, j) is called *incident* to both the vertices i and j which are neighbors. A *self-loop* is an edge that join a vertex i to itself. A *multi-edge* is a collection of two or more edges having identical endpoints. A *simple graph* is a graph which has neither self-loops nor multi-edges.

For a simple graph with vertex set V, the *adjacency matrix* A is a square $V \times V$ matrix A such that its element A_{ij} is one when there is an edge from vertex i to vertex j, and zero when there is no edge. That is,

$$A_{ij} = \begin{cases} 1, & \text{if } (i, j) \in E, \\ 0, & \text{otherwise.} \end{cases}$$

The matrix representation of a graph is very useful because it allows us apply the graph using matrix theory and linear algebra. A graph $G(V, E)$ associated with the adjacency matrix A is denoted by $G_A(V, E)$. This is for the graphs with only non-negative edges. However, there exist graphs with negative edges, which are called *signed graphs* [124], [107], and [108]. A signed adjacency matrix is defined as

$$A_{ij} = \begin{cases} -1, & \text{if } (i,j) \in E^-, \\ 1, & \text{if } (i,j) \in E^+, \\ 0, & \text{otherwise,} \end{cases}$$

where E^- is the collection of the negative edges, E^+ is the collection of the positive edges, and $E^- \cap E^+ = \emptyset$. A signed graph $G(V, E^-, E^+)$ associated with the adjacency matrix A is denoted by $G_A(V, E^-, E^+)$.

A graph is called *undirected* if the edges have no direction, i.e., each pair $(i, j) \in E$ is considered unordered and $(i, j) = (j, i)$, such as in friendship networks (where each relationship is considered reciprocal). The adjacency matrix associated to an undirected graph is symmetric, $A = A^T$. If the direction of the edges matters, the graph is called *directed* and an edge (i, j) has a source i and a destination j. Telecommunication networks are a good example of directed network where one makes a distinction between the caller and the callee. In directed graphs, a neighbor j of a node i is called a *child* when $(i, j) \in E$ and a *parent* when $(j, i) \in E$.

The number of neighbors of a node i is called the *degree*, denoted by k_i. In directed networks, it is natural to distinguish between the *in-degree* k_i^{in} and the *out-degree* k_i^{out} as the number of parents and the number of children, respectively. Using the adjacency matrix A, the vectors of in and out-degrees can be computed as

$$k^{in} = A^T \mathbf{1}, \; k^{out} = A\mathbf{1},$$

where $\mathbf{1}$ is the vector of all 1's of length $\mid V \mid$.

Besides the direction, one can associate a weight to each edge representing the intensity of the interaction between the incident vertices, i.e., (i, j, w_{ij}), and the graph is then called *weighted*. Weighted graphs can be represented by their weighted adjacency matrix $W \in \mathbb{R}^{n \times n}$ such that $W_{i,j} \neq 0$ if and only if $A_{ij} = 1$. We define the *strength* $s_i$of node i as the sum of the weights of its incident edges, similarly to the degree. If the graph is also directed, we make a distinction between the *in-strength* and the *out-strength* as the sum of the weights of the incoming and outgoing edges,

respectively. The vectors of in and out-strengths can be represented using the weighted adjacency matrix W as

$$s^{in} = W^T\mathbf{1},\ s^{out} = W\mathbf{1}.$$

The *density* of a graph is defined as the ratio between the actual number of edges in the graph and the maximal number of possible edges, i.e., $\frac{m}{n^2}$ for a directed graph with self-loops. A graph is called *sparse* if its density is low, which implies that its adjacency matrix is also sparse. The maximum number of edges being quadratic in the number of nodes in the graph, it is generally assumed that the number of edges in sparse graphs should grow linearly with the number of nodes, $m = \mathcal{O}(n)$.

C.1.1 Graph Structures

We define some structural properties of graphs in this section.

A *complete* graph is a simple undirected graph in which every pair of distinct vertices is joined by a unique edge. A *complete digraph* is a directed graph in which every pair of distinct vertices is connected by a pair of unique edges (one in each direction).

A *subgraph* $H(V_H, E_H)$ of a graph $G(V, E)$ is a graph whose vertices are a subset of the vertices of G, $V_H \subset V$ and whose edges are a subset of the edges of G incident only to nodes in V_H, $E_H \subset \{(i,j) \mid i,j \in V_H, (i,j) \in E\}$. A subgraph is called *induced* if the edge set E_H contains edges that have both endpoints in V_H. A *spanning graph* is a subgraph that has the same node set as the original graph, i.e., $V_H = V$.

We call an undirected graph as *connected*, if there exists a path between any pair of nodes. If a graph is *disconnected*, then we can divide it into multiple induced subgraphs $H_i(V_{H_i}, E_{H_i})$ such that every subgraph is connected and all the edges of the original graph are contained in one of the subgraphs, i.e., there is no edge between any of the induced subgraphs. Each induced subgraphs H_i is called a *connected component* of the original graph G. If the graph is directed, there are two different types of connected components. The first type is a *strongly connected component* (SCC) that is a maximal set of vertices such that there exists a directed path between every pair of vertices. Strongly connected components are simply connected components in an undirected graph. The second type is a *weakly connected component* (WCC) that is a maximal set of vertices such that there exists a path between every pair of vertices in the associated undirected induced graph.

A *clique* is a subset of vertices of an undirected graph such that its induced subgraph is complete. A *maximal clique* is a clique that cannot be extended by including one more adjacent vertex, that is, a clique which does not exist exclusively within the vertex set of a larger clique. One of classical methods to detect overlapping communities, called the clique percolation method [88], builds up the communities from k-cliques.

The last notion we would like to mention is the *complement* of a graph $G(V, E)$, which is a graph $G^{\perp}(V, E^{\perp})$ defined over the same set of nodes but such that $E^{\perp}$ contains all the edges not existing in G, i.e., $E^{\perp} = \{(i, j) \mid (i, j) \notin E\}$.

C.1.2 Random Graphs

To assess the quality of algorithms an measures, we extensively use random graphs in this dissertation. A graph is called *random*, if either its vertex set, its edge set or both are generated using a random process. Random graphs are very useful to model typical properties of networks or to quantify how similar or dissimilar is a given network from what one would observe on average in networks having similar properties. Erdos and Renyi model [36] and configuration null model [83] are two common random graphs models.

C.1.3 Erdos and Renyi (ER) Model

Erdos and Renyi model is the most common and simplest random graph model. To build such a random graph, one begins with a fixed set of vertices and no edges. Then, one adds the $n(n-1)$ possible edges (excluding self-loops) independently with a constant probability $p \in [0, 1]$. Independence means that the probability that a pair of vertices is connected by an edge is independent from the presence or absence of edges among other pairs. Each pair of vertices can be regarded as a random binary variable, taking the value 1 (edge present) or 0 (edge absent). The random graph is equivalent to a series of independent random binary variables because each vertex pair is assigned an edge with the same probability and independent of all other pairs. Such a series is called a Bernoulli process and for this reason the model is also known as the *Bernoulli random graph model* [30].

The expected number of edges in an undirected Erdos-Renyi graph is $\langle m \rangle = \binom{n}{2}p$, where $\binom{n}{2}$ is the binomial coefficient, that is, the number of possible ways to select 2 nodes out of a set of n nodes. The expected number of edges is simply multiplied by 2 for directed Erdos-Renyi graphs.

The degree distribution of a network is the probability distribution of the degree of the vertices in the network. More precisely, the degree distribution $P(k)$ is the probability that a vertex taken at random has a degree k and is given by the expected proportion of vertices of degree k in a random network

$$P(k) = \frac{\langle | \{i \mid k_i = k\} | \rangle}{n}.$$

In Erdos-Renyi graphs, the degree distribution is binomial

$$P(k) = \binom{n-1}{k} p^k (1-p)^{n-1-k}. \tag{C.1}$$

For large n and small p, the degree distribution (C.1) can be well approximated by a Poisson distribution

$$P(k) \approx \frac{(np)^k e^{-np}}{k!}. \tag{C.2}$$

The random null model is often used as a term of comparison, to verify whether the original graph in question displays some structural features. We want that p is chosen such that the original graph and the null model have on average the same number of edges. So $m = p\binom{n}{2}$, hence we set $p = \frac{m}{\binom{n}{2}}$, where m is the number of edges and n is the number of nodes.

The Erdos-Renyi null model has been widely used to create random graphs, since it is easy to be constructed. However, it is not straightforward to extend the model to weighted graphs. While $p = \frac{m}{n^2}$ gives the probability to connect any pair of nodes such that the expected number of edges in the ER null model is m, defining the weight of such edges would require additional and specific knowledge about the network [19].

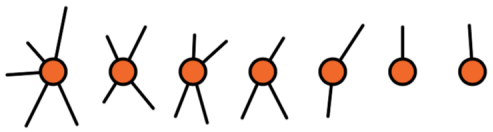
In addition, the degree distribution of Erdos-Renyi random graph follows a Poisson distribution. In contrast, most real-world graphs exhibit heavy-tailed power degree distributions [96]. This indicates that the number of edges might not be the best feature to match in the original graph. We can improve this aspect by using the same degree distribution as the original network [71] [77], which is called the configuration model.

C.1.4 Configuration Null (CNM) Model

The configuration null model uses the same degree distribution as the original network to match the original graph [71] [77]. *Configuration null model* takes the degree $k_i = \sum_j A_{ij}$ of a vertex into

account, which is constructed with the same degree distribution as the original network, where A is the adjacency matrix of the original network.

Let us first consider an unweighted undirected network and explain how to build a random graph using the configuration model. We can cut all links in half, so that each node i has k_i "stubs", and connect all the stubs randomly. The central mathematical property of the configuration model is the probability p_{ij} that two vertices i, j are connected. Under a random matching on the edge stubs,



for a particular stub attached to vertex i, there are k_j possible stubs out of $2m-1$ (excluding the stub on i under consideration), attached to j to which it could connect. And there are k_i chances that this could happen. So we can get $p_{ij} = \frac{k_i k_j}{2m-1} \simeq \frac{k_i k_j}{2m}$, where the second form holds in the limit of large m.

One can observe that the configuration null model matches , on average, both the number of edges in the original network and the degree sequence.

We can make a straightforward extension to weighted directed graphs based on the creation for an unweighted undirected network using the outgoing and incoming strengths instead of the degrees for the null model. The probability $p_{ij}^{w,d}$ that two vertices i, j are connected for a weighted directed graph is

$$p_{ij}^{w,d} = \frac{s_i^{out} s_j^{in}}{m_w}, \tag{C.3}$$

where the strength s_i^{out} of vertex i is defined as the sum of weights of the outgoing edges connected to it, and the strength s_i^{in} of vertex i is defined as the sum of weights of the incoming edges connected to it, and m_w is the total weights of the original network.

C.2 Communites in Network

Community structure [84] is the division of network nodes into groups within which the network connections are dense, but between which are sparser. These groups are called *communities*, or *modules*. A figurative sketch of a network with such a community structure is shown in Figure C.1.

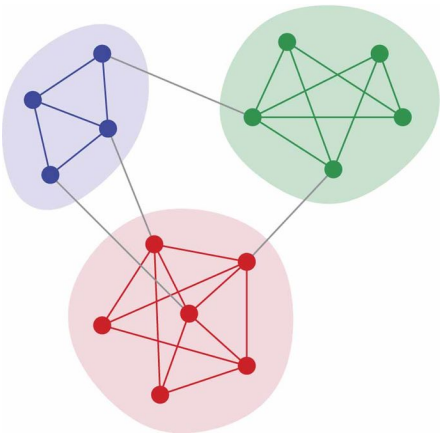


Figure C.1: A small network with 3 communities, which have dense internal links but between which there is only a lower density of external links.

C.3 Quality Functions for Community Detection Problem

C.3.1 GN modularity

In real life, the algorithms normally are used on networks for which the communities are not known ahead of time. How do we know when the communities found by the algorithms are good ones? To answer these questions, Newman and Girvan [84] define a measure of the quality of a particular division of a network, which is called the modularity.

To define the modularity, they consider a particular division of a network into k communities. Then they define a $k \times k$ symmetric matrix e whose element e_{cd} is the fraction of links that link nodes in community c to community d. Here we always consider all edges in the original network even after we remove edges later. The trace of this matrix Tr $e = \sum_c e_{cc}$ gives the fraction of edges in the network that connect vertices in the same community. Clearly a good division should have a high value of this trace, but the trace itself is not a good measure of the quality of the division, since Tr e get the maximal value 1 when we place all vertices in a single community, which gives us no information about the community structure at all.

So Newman and Girvan further define the row sums $a_c = \sum_d e_{cd}$, which represent the fraction of edges that connect to vertices in community c. And they define the modularity as follows, which measures the fraction of the edges in the network that connect vertices of the same type (i.e., within-community edges) minus the expected value of the same quantity in a network with the

same community divisions but random connections between the vertices.

$$Q = \sum_c e_{cc} - a_c^2, \quad \text{with } a_c = \sum_d e_{cd}. \tag{C.4}$$

Clauset et al. [27] give an equivalent formula with formula (C.4) for GN modularity, which is

$$Q = \frac{1}{2m} \sum_{i,j} (A_{ij} - \frac{k_i k_j}{2m}) \delta(\sigma_i, \sigma_j) \tag{C.5}$$

$$= \frac{1}{2m} \sum_c \sum_{i,j} (A_{ij} - \frac{k_i k_j}{2m}) \delta(\sigma_i, c) \delta(\sigma_j, c) \tag{C.6}$$

$$= \frac{1}{2m} \sum_c [2\Sigma_{in} - \frac{(\Sigma_{tot})^2}{2m}] \tag{C.7}$$

$$= \sum_c [\frac{\Sigma_{in}}{m} - (\frac{\Sigma_{tot}}{2m})^2], \tag{C.8}$$

where i and j are two vertices, A_{ij} is an element of the adjacency matrix with $A_{ij} = 1$ if there is an edge (i, j) and zero otherwise; where $k_i = \sum_j A_{ij}$ is the degree of vertex i and m is the total number of edges in the network, $\delta(\sigma_i, \sigma_j)$ is the Kronecker delta symbol so that $\delta(\sigma_i, \sigma_j) = 1$ if $\sigma_i = \sigma_j$ both i and j are in the same community, and σ_i is the label of the community to which vertex i is assigned. This is called the classical modularity. In equation(C.8), Σ_{in} is the sum of the numbers of the links inside community c, and Σ_{tot} is the sum of the numbers of the links incident to nodes in c. From this, we can see that the modularity is the fraction of edges within communities minus the expected fraction of such edges, which is corresponding to formula(C.4).

C.3.2 A General Framework of Quality Functions

From the definition of communities, we know that communities are groups of densely interconnected nodes that are only sparsely connected with the rest of the network. From this, Reichardt and Bornholdt [94] find four requirements of the quality function: it should

1. reward a_{ij} to existing edges in the same group
2. penalize b_{ij} to missing edges in the same group
3. penalize c_{ij} to existing edges between different groups
4. reward d_{ij} to missing edges between different groups.

This leads to the following function which is closely related to the modularity from a more general framework:

$$\begin{aligned}\mathscr{H}(\sigma) = &-\sum_{i,j}[a_{ij}A_{ij} - b_{ij}(1 - A_{ij})]\delta(\sigma_i, \sigma_j) \\ &+ [-c_{ij}A_{ij} + d_{ij}(1 - A_{ij})](1 - \delta(\sigma_i, \sigma_j)),\end{aligned} \tag{C.9}$$

where i and j are two vertices, A_{ij} is an element of the adjacency matrix with $A_{ij} = 1$ if there is an edge (i, j) and zero otherwise, $\delta(\sigma_i, \sigma_j)$ is the Kronecker delta symbol so that $\delta(\sigma_i, \sigma_j) = 1$ if $\sigma_i = \sigma_j$ both i and j are in the same community, and σ_i is the label of the community to which vertex i is assigned.

The negative sign is only a matter of convention, so we want to minimize this function. So the optimization problem [106] is

$$\min_\sigma \mathscr{H}(\sigma) \tag{C.10}$$

over all possible partitions.

So $\mathscr{H}(\sigma)$ is the cost of a partition, and the optimal partition has the minimal cost.

We can simplify the function $\mathscr{H}(\sigma)$ as follows if we suppose the edges within communities are equally rewarded (or punished) as edges between communities, i.e., $a_{ij} = c_{ij}$ and $b_{ij} = d_{ij}$, and remove factors which do not depend on σ.

$$\mathscr{H}(\sigma) = -\sum_{i,j}(a_{ij}A_{ij} - b_{ij}(1 - A_{ij}))\delta(\sigma_i, \sigma_j). \tag{C.11}$$

The weights a_{ij}, b_{ij} are non-negative, and remain to be specified.

Reichardt and Bornholdt Model. Reichardt and Bornholdt [94] compared the original network to a random network, a random null model. The null model is a graph which matches one specific graph in some of its structural features, but which is otherwise taken to be an instance of a random graph. The null model is used as a term of comparison, to verify whether the graph in question displays some feature, such as community structure. We assume the random null model does not have the community structure.

Assume p_{ij} is the probability for the (i, j) link, which be specified later. If we take $a_{ij} = w_{ij} - \gamma_{RB}p_{ij}$, $b_{ij} = \gamma_{RB}p_{ij}$, where w_{ij} is the weight of the (i, j) link, γ_{RB} is used to weight the importance of the random network, then we can get the Reichardt and Bornholdt cost function

$$\mathscr{H}_{RB} = -\sum_{i,j}(w_{ij}A_{ij} - \gamma_{RB}p_{ij})\delta(\sigma_i, \sigma_j) \tag{C.12}$$

If we assume the graph is unweighted and undirected, so $w_{ij} = 1$. We can rewrite the RB cost function as

$$\mathscr{H}_{rb} = -\sum_{i,j}(A_{ij} - \gamma_{RB}p_{ij})\delta(\sigma_i, \sigma_j) \tag{C.13}$$

$$= -\sum_{c}\sum_{i,j}(A_{ij} - \gamma_{RB}p_{ij})\delta(\sigma_i, c)\delta(\sigma_j, c) \tag{C.14}$$

$$= -\sum_{c}[\sum_{i,j} A_{ij}\delta(\sigma_i, c)\delta(\sigma_j, c) - \gamma_{RB}p_{ij}\delta(\sigma_i, c)\delta(\sigma_j, c)] \tag{C.15}$$

$$:= -\sum_{c}[e_c - \gamma_{RB}\langle e_c\rangle_{p_{ij}}] \tag{C.16}$$

where $e_c = \sum_{i,j} A_{ij}\delta(\sigma_i, c)\delta(\sigma_j, c)$ is the number of edges in community c; $\langle e_c\rangle = \sum_{i,j} p_{ij}\delta(\sigma_i, c)\delta(\sigma_j, c)$ is the expected number of edges in community c.

Remark C.3.1 (Extension to weighted undirected networks)**.** *For the weighted undirected networks, $e_c^w = \sum_{i,j} w_{ij}A_{ij}\delta(\sigma_i, c)\delta(\sigma_j, c)$ is the sum of weights of edges inside the community c; $\langle e_c^w\rangle = \sum_{i,j} p_{ij}^w\delta(\sigma_i, c)\delta(\sigma_j, c)$ is the expected sum of weights of edges in community c. For simplicity, we consider the unweighted undirected networks firstly, then we extend them to weighted networks later.*

The specific form of cost function depends on the null models we choose. Various random null models can be chosen, that is, various p_{ij} can be chosen. The null model should match the original graph for some of its structural features, but is essentially random.

- RB model with Erdos-Renyi null model:

  For ER null model, we choose p as a constant, and we want that p is chosen such that the original graph and the null model have on average the same number of edges. So $m = p\binom{n}{2}$, hence we set $p_{ij} = p = \frac{m}{\binom{n}{2}}$, where m is the number of edges and n is the number of nodes. Then the expected number of edges within a community is pn_c^2, where n_c is the the number of nodes of community c. So the RB model with Erdos-Renyi null model is

  $$\mathscr{H}_{rb} = -\sum_{c}(e_c - \gamma_{RB}pn_c^2) \tag{C.17}$$

  where $e_c = \sum_{i,j} A_{ij}\delta(\sigma_i, c)\delta(\sigma_j, c)$ is the number of edges in community c for unweighted undirected networks.

  The Erdos-Renyi null model has been widely used to create random graphs, since it is easy to be constructed. However, it is not straightforward to extend the model to weighted graphs.

While $p_{ij} = p = \frac{m}{n^2}$ gives the probability to connect any pair of nodes such that the expected number of edges in the ER null model is m, defining the weight of such edges would require additional and specific knowledge about the network [19]. But if we can define the weights properly, then the RB model with Erdos-Renyi null model for weighted netwoks is

$$\mathscr{H}_{RB} = -\sum_c (e_c^w - \gamma_{RB} p n_c^2) \tag{C.18}$$

where $e_c^w = \sum_{i,j} w_{ij} A_{ij} \delta(\sigma_i, c)\delta(\sigma_j, c)$ is the sum of weights of edges inside the community c.

In addition, the degree distribution of ER random graph can be showed that it follows a Poisson distribution. In contrast, most real-world graphs exhibit heavy-tailed degree distributions. We can improve this aspect of our random null graph model by using the following configuration model $G(n, \overrightarrow{k})$, where $\overrightarrow{k} = \{k_i\}$ is a degree sequence and k_i is the degree of vertex i.

- RB model with configuration null model:

  The precise mathematical properties for a configuration random model depend on the degree sequence. Configuration null model takes the degree $k_i = \sum_j A_{ij}$ of a node into account, which is constructed with the same degree distribution as the original network. We cut all links in half, so that each node i has k_i "stubs", and connect all the stubs randomly. The central mathematical property of the configuration model (and indeed, all random graph models) is the probability p_{ij} that two vertices i, j are connected.

  Under a random matching on the edge stubs, for a particular stub attached to vertex i, there are k_j possible stubs out of $2m - 1$ (excluding the stub on i under consideration), attached to j to which it could connect. And there are k_i chances that this could happen. So we can get $p_{ij} = \frac{k_i k_j}{2m-1} \simeq \frac{k_i k_j}{2m}$, where the second form holds in the limit of large m.

$$\mathscr{H}_{rb} = -\sum_{i,j} (A_{ij} - \gamma_{RB} \frac{k_i k_j}{2m}) \delta(\sigma_i, \sigma_j) \tag{C.19}$$

$$= -\sum_c [e_c - \gamma_{RB} \frac{K_c^2}{2m}], \tag{C.20}$$

  where $e_c = \sum_{i,j} A_{ij} \delta(\sigma_i, c)\delta(\sigma_j, c)$ is the number of edges in community c; $K_c = \sum_i k_i \delta(\sigma_i, c)$ is the sum of degrees of the nodes in community c.

  If we take $\gamma_{RB} = 1$, we can get the relation between RB model with configuration null model and the classical modularity Q, which is $Q = -\frac{1}{2m}\mathscr{H}_{rb}$. So maximizing the modularity of a community structure is equivalent to minimizing the cost function of RB model with configuration null model.

  The configuration null model is straightforward to extend to weighted networks: we use strengths of nodes instead of degrees of nodes for weighted undirected networks. In weighted

networks, the strength s_i of node i is defined as the sum of weights of the edges connected to it. So $e_c^w = \sum_{i,j} w_{ij} A_{ij} \delta(\sigma_i, c)\delta(\sigma_j, c)$ and $\langle e_c^w \rangle = \frac{S_c^2}{2S}$, where $S_c = \sum_i s_i \delta(\sigma_i, c)$ is the sum of strengths of the nodes in community c, $S = \sum_{i=1}^n s_i$ is the total strength of the network. Therefore,

$$\mathscr{H}_{RB} = -\sum_c (e_c^w - \gamma_{RB}\langle e_c^w \rangle) \tag{C.21}$$

$$= -\sum_c [e_c^w - \gamma_{RB}\frac{S_c^2}{2S}]. \tag{C.22}$$

Null Model. The classical modularity (RB model in general) actually has a problem, which is the resolution limit. Fortunato et.al find that modularity contains an intrinsic scale of order $\sqrt{m}$, which constraints the number and the size of the modules, where m is the total number of links [41]. Modules that are smaller than this scale might bot be resolved. Due to the resolution limit, some small communities in large graphs may not be detected, and we can see the problem from the following Figure [109].

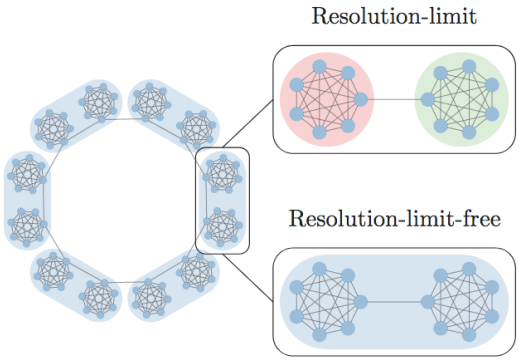


Figure C.2: Resolution Limit Problem of the classical modularity, see [109].

In Figure C.2, we can see the ring of cliques which are connected through single links. So each clique should represent a community, but modularity may merge them. The RB model does have the resolution limit problem. The RB model with a configuration null model for example merges two neighboring cliques in this ring network of cliques when $\gamma_{RB} < \frac{q}{n_c(n_c-1)+2}$, where q is the number of cliques and n_c is the number of nodes of a clique. Since the number of cliques q is a global variable, it shows modularity might be "hiding" some smaller communities within larger communities, depending on the size of the network.

To avoid the resolution limit problem, some no null models are developed. Here, we just consider Ronhovde and Nussinov (RN) model [97] and Constant Potts Model (CPM) model [109].

- RN model:

  For the general modularity equation (C.11), if we take $a_{ij} = w_{ij}, b_{ij} = \gamma_{RN}$, then we can get the RN model [97]. The cost function of RN model is

  $$\mathscr{H}_{RN}(\sigma) = -\sum_{i,j}(A_{ij}(w_{ij} + \gamma_{RN}) - \gamma_{RN})\delta(\sigma_i, \sigma_j). \tag{C.23}$$

  We can see that this model has no null model. The RN model only join two cliques when $\gamma_{RN} < \frac{1}{n_c^2 - 1}$ [97], which does not depend on the number of cliques q, and depends only on the local variable n_c, so is argued not to suffer from any resolution limit.

- Constant Potts Model:

  We can get the CPM model [109], if we take $a_{ij} = w_{ij} - b_{ij}, b_{ij} = \gamma_{CPM}$ in the general modularity equation (C.11). The cost function of CPM model is

  $$\mathscr{H}_{CPM}(\sigma) = -\sum_{i,j}(A_{ij}w_{ij} - \gamma_{CPM})\delta(\sigma_i, \sigma_j). \tag{C.24}$$

  We can find that this model only compares the original network to a constant parameter, not a random network, so this model has no null model. CPM merges two neighboring cliques in this ring network of cliques when $\gamma_{RB} < \frac{1}{n_c^2}$ [109], which also does not depend on the number of cliques q and can hence also avoid the resolution limit problem.

BIBLIOGRAPHY

[1] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, Princeton, NJ, 2008.

[2] P.-A. Absil and J. Malick. Projection-like retractions on matrix manifolds. *SIAM Journal on Optimization*, 22(1):135–158, 2012.

[3] Fevzi Alimoğlu and Ethem Alpaydin. Combining multiple representations for pen-based handwritten digit recognition. *Turkish Journal of Electrical Engineering and Computer Sciences*, 9(1):1–12, 2001.

[4] Arthur Asuncion and David Newman. Uci machine learning repository, 2007.

[5] Franz Aurenhammer, Rolf Klein, and Der-Tsai Lee. *Voronoi Diagrams and Delaunay Triangulations*. World Scientific Publishing Company, 2013.

[6] Francis R. Bach and Michael I. Jordan. Learning spectral clustering. In *Proceedings of the 16th International Conference on Neural Information Processing Systems*, NIPS'03, page 305–312, Cambridge, MA, USA, 2003. MIT Press.

[7] C. G. Baker. *Riemannian manifold trust-region methods with applications to eigenproblems*. PhD thesis, Florida State University, Department of Computational Science, 2008.

[8] Amir Beck. *First-order methods in optimization*. SIAM, 2017.

[9] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[10] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[11] Dimitri P Bertsekas, WW Hager, and OL Mangasarian. *Nonlinear programming*. Athena Scientific Belmont, MA, 1998.

[12] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[13] Nicolas Boumal. *An introduction to optimization on smooth manifolds*. Cambridge University Press, 2023.

[14] Gergana Bounova. Matlab Tools for Network Analysis. `http://strategic.mit.edu/downloads.php?page=matlab_networks`, 2009.

[15] Christos Boutsidis, Petros Drineas, and Michael W Mahoney. Unsupervised feature selection for the k-means clustering problem. In *Advances in Neural Information Processing Systems*, pages 153–161, 2009.

[16] Paul S Bradley and Usama M Fayyad. Refining initial points for k-means clustering. In *ICML*, volume 98, pages 91–99. Citeseer, 1998.

[17] Ulrik Brandes. A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology*, 25(2):163–177, 2001.

[18] Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Gorke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. On modularity clustering. *IEEE Transactions on Knowledge and Data Engineering*, 20(2):172–188, 2007.

[19] Arnaud Browet. *Algorithms for community and role detection in networks.* PhD thesis, Catholic University of Louvain, Louvain-la-Neuve, Belgium, 2014.

[20] Timothy Carson, Dustin G Mixon, and Soledad Villar. Manifold optimization for k-means clustering. In *2017 International Conference on Sampling Theory and Applications (SampTA)*, pages 73–77. IEEE, 2017.

[21] M Emre Celebi, Hassan A Kingravi, and Patricio A Vela. A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert Systems with Applications*, 40(1):200–210, 2013.

[22] P. K Chan and F Schlag. Spectral k -way ratio-cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(9):P.1088–1096, 1994.

[23] Feiyu Chen, Yuchen Yang, Liwei Xu, Taiping Zhang, and Yin Zhang. Big-data clustering: K-means or k-indicators? *arXiv Preprint arXiv:1906.00938*, 2019.

[24] Mo Chen. Pattern recognition and machine learning toolbox. *MATLAB Central File Exchange*, 2021.

[25] Shixiang Chen, Shiqian Ma, Anthony Man-Cho So, and Tong Zhang. Proximal gradient method for nonsmooth optimization over the Stiefel manifold. *SIAM Journal on Optimization*, 30(1):210–239, 2020.

[26] F. H. Clarke. *Optimization and nonsmooth analysis.* Classics in Applied Mathematics of SIAM, 1990.

[27] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical Review E*, 70(6):066111, 2004.

[28] Leon Danon, Albert Diaz-Guilera, and Alex Arenas. The effect of size heterogeneity on community identification in complex networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2006(11):P11010, 2006.

[29] Leon Danon, Albert Diaz-Guilera, Jordi Duch, and Alex Arenas. Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(09):P09008, 2005.

[30] Wouter De Nooy, Andrej Mrvar, and Vladimir Batagelj. *Exploratory social network analysis with Pajek: Revised and expanded edition for updated software*, volume 46. Cambridge University Press, 2018.

[31] Inderjit Dhillon, Yuqiang Guan, and Brian Kulis. A unified view of kernel k-means. spectral clustering and graph cuts. *Technical Report, Department of Computer Sciences, University of Texas at Austin*, 2005.

[32] Inderjit S Dhillon and Dharmendra S Modha. Concept decompositions for large sparse text data using clustering. *Machine learning*, 42:143–175, 2001.

[33] Chris Ding and Tao Li. Adaptive dimension reduction using discriminant analysis and k-means clustering. In *Proceedings of the 24th International Conference on Machine Learning*, pages 521–528, 2007.

[34] M. P. do Carmo. *Riemannian geometry*. Mathematics: Theory & Applications, 1992.

[35] A. Edelman, T. A. Arias, and S. T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications*, 20(2):303–353, January 1998. doi:10.1137/S0895479895290954.

[36] Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

[37] Maurizio Filippone, Francesco Camastra, Francesco Masulli, and Stefano Rovetta. A survey of kernel and spectral methods for clustering. *Pattern Recognition*, 41(1):176–190, 2008.

[38] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

[39] Edward W Forgy. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *Biometrics*, 21:768–769, 1965.

[40] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3-5):75–174, 2010.

[41] Santo Fortunato and Marc Barthelemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1):36–41, 2007.

[42] Santo Fortunato and Claudio Castellano. Community structure in graphs. In *Computational Complexity*, pages 490–512. Springer, 2012.

[43] Linton C Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41, 1977.

[44] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

[45] Jonathan L Gross and Jay Yellen. *Graph theory and its applications*. CRC press, 2005.

[46] Greg Hamerly and Charles Elkan. Alternatives to the k-means algorithm that find better clusterings. In *Proceedings of the Eleventh International Conference on Information and knowledge Management*, pages 600–607, 2002.

[47] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[48] Robert Hermann, Jeff Cheeger, and David G. Ebin. Comparison theorems in Riemannian geometry. *Bulletin of the American Mathematical Society*, 82(6):834–836, 1976.

[49] W. Huang and K. Wei. Riemannian proximal gradient methods. *Mathematical Programming*, 2021. published online, DOI:10.1007/s10107-021-01632-3.

[50] Wen Huang. *Optimization algorithms on Riemannian manifolds with applications*. PhD thesis, The Florida State University, 2013.

[51] Wen Huang, P-A Absil, and Kyle A Gallivan. Intrinsic representation of tangent vectors and vector transports on matrix manifolds. *Numerische Mathematik*, 136(2):523–543, 2017.

[52] Wen Huang, P-A Absil, Kyle A Gallivan, and Paul Hand. Roptlib: an object-oriented c++ library for optimization on Riemannian manifolds. *ACM Transactions on Mathematical Software (TOMS)*, 44(4):43, 2018.

[53] Wen Huang and Ke Wei. An extension of fast iterative shrinkage-thresholding algorithm to Riemannian optimization for sparse principal component analysis. *Numerical Linear Algebra with Applications*, page e2409, 2021.

[54] Wen Huang and Ke Wei. An inexact Riemannian proximal gradient method. *Computational Optimization and Applications*, pages 1–32, 2023.

[55] Jonathan J. Hull. A database for handwritten text recognition research. *IEEE Transactions on pattern analysis and machine intelligence*, 16(5):550–554, 1994.

[56] Svante Janson, Tomasz Luczak, and V Kolchin. Random graphs. *Bulletin of the London Mathematical Society*, 33:363–383, 2001.

[57] Ian T Jolliffe, Nickolay T Trendafilov, and Mudassir Uddin. A modified principal component technique based on the lasso. *Journal of Computational and Graphical Statistics*, 12(3):531–547, 2003.

[58] Roy Jonker and Ton Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. In *DGOR/NSOR: Papers of the 16th Annual Meeting of DGOR in Cooperation with NSOR/Vorträge der 16. Jahrestagung der DGOR zusammen mit der NSOR*, pages 622–622. Springer, 1988.

[59] Zhao Kang, Chong Peng, Qiang Cheng, and Zenglin Xu. Unified spectral clustering with optimal graph. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[60] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.

[61] Athanasios Kehagias. Community Detection Toolbox. `https://www.mathworks.com/matlabcentral/fileexchange/45867-community-detection-toolbox`, 2021.

[62] Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[63] Harold W Kuhn. Variants of the hungarian method for assignment problems. *Naval Research Logistics Quarterly*, 3(4):253–258, 1956.

[64] Andrea Lancichinetti and Santo Fortunato. Community detection algorithms: a comparative analysis. *Physical Review E*, 80(5):056117, 2009.

[65] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Physical Review E*, 78(4):046110, 2008.

[66] John M Lee. *Riemannian manifolds: an introduction to curvature*, volume 176. Springer Science & Business Media, 2006.

[67] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):2–es, 2007.

[68] X. Li, D. Sun, and K.-C. Toh. A highly efficient semismooth Newton augmented Lagrangian method for solving Lasso problems. *SIAM Journal on Optimization*, 28(1):433–458, 2018.

[69] Xudong Li, Defeng Sun, and Kim-Chuan Toh. A highly efficient semismooth newton augmented lagrangian method for solving lasso problems. *SIAM Journal on Optimization*, 28(1):433–458, 2018.

[70] Stuart Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[71] Tomasz Luczak. Sparse random graphs with a given degree sequence. In *Proceedings of the Symposium on Random Graphs, Poznan*, pages 165–182, 1989.

[72] David JC MacKay and David JC Mac Kay. *Information theory, inference and learning algorithms*. Cambridge University Press, 2003.

[73] Andrzej Manitius. Optimization and nonsmooth analysis (frank h. clarke). *SIAM Review*, 27(2):288–291, 1985.

[74] Christopher D Manning and Prabhakar Raghavan. utze, Introduction to information retrieval, 2008.

[75] Melissa Sue Marchand. *Low-Rank Riemannian Optimization Approach to the Role Extraction Problem*. PhD thesis, The Florida State University, 2017.

[76] EM Mirkes. K-means and k-medoids applet. *University of Leicester*, 2011.

[77] Michael Molloy and Bruce Reed. A critical point for random graphs with a given degree sequence. *Random Structures & Algorithms*, 6(2-3):161–180, 1995.

[78] Mark EJ Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167–256, 2003.

[79] Mark EJ Newman. Analysis of weighted networks. *Physical review E*, 70(5):056131, 2004.

[80] Mark EJ Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69(6):066133, 2004.

[81] Mark EJ Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74(3):036104, 2006.

[82] Mark EJ Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[83] Mark EJ Newman. *Networks: an introduction*. Oxford University Press, Oxford, 2010.

[84] Mark EJ Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2):026113, 2004.

[85] Mark EJ Newman and Elizabeth A Leicht. Mixture models and exploratory analysis in networks. *Proceedings of the National Academy of Sciences*, 104(23):9564–9569, 2007.

[86] Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, pages 849–856, 2002.

[87] Barrett O'neill. *Semi-Riemannian geometry with applications to relativity*. Academic Press, 1983.

[88] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *nature*, 435(7043):814–818, 2005.

[89] Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):127–239, 2014.

[90] Filippo Pompili, Nicolas Gillis, P-A Absil, and François Glineur. Two algorithms for orthogonal nonnegative matrix factorization with application to clustering. *Neurocomputing*, 141:15–25, 2014.

[91] Chunhong Qi. *Numerical optimization methods on Riemannian manifolds*. PhD thesis, The Florida State University, 2011.

[92] Jörg Reichardt. Stefan bornholdt: Statistical mechanics of community detection. *Physical Review E*, 74:1–14, 2006.

[93] Jörg Reichardt. *Structure in Complex Networks*, volume 766. Springer, 2008.

[94] Jörg Reichardt and Stefan Bornholdt. Statistical mechanics of community detection. *Physical Review E*, 74(1):016110, 2006.

[95] Jörg Reichardt and Douglas R White. Role models for complex networks. *The European Physical Journal B*, 60(2):217–224, 2007.

[96] Albert Reka. Barabási. *Statistical mechanics of complex networks," Rev. Mod. Phys*, 74:47–97, 2002.

[97] Peter Ronhovde and Zohar Nussinov. Local resolution-limit-free potts model for community detection. *Physical Review E*, 81(4):046114, 2010.

[98] Martin Rosvall and Carl T Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123, 2008.

[99] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

[100] Antoine Scherrer. Matlab Version for Louvain's Algorithm. `https://perso.uclouvain.be/vincent.blondel/research/louvain.html`, 2008.

[101] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

[102] Claude E Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[103] Jianbo Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[104] Pedro FB Silva, Andre RS Marcal, and Rubim M Silva. Evaluation of features for leaf discrimination. In *International Conference Image Analysis and Recognition*, pages 197–204. Springer, 2013.

[105] Jianbo Shi Timothee Cour, Stella Yu. Normalized cut segmentation code. *Copyright 2004 University of Pennsylvania, Computer and Information Science Department*, 2004.

[106] Vincent Traag. *Algorithms and Dynamical Models for Communities and Reputation in Social Networks*. Springer, 2014.

[107] Vincent A Traag and Jeroen Bruggeman. Community detection in networks with positive and negative links. *Physical Review E*, 80(3):036115, 2009.

[108] Vincent A Traag, Gautier Krings, and Paul Van Dooren. Significant scales in community structure. *Scientific Reports*, 3(1):1–10, 2013.

[109] Vincent A Traag, Paul Van Dooren, and Yurii Nesterov. Narrow scope for resolution-limit-free community detection. *Physical Review E*, 84(1):016114, 2011.

[110] Vincent A Traag, Ludo Waltman, and Nees Jan Van Eck. From louvain to leiden: guaranteeing well-connected communities. *Scientific Reports*, 9(1):5233, 2019.

[111] P Van Mieghem, X Ge, P Schumm, S Trajanovski, and H Wang. Spectral graph analysis of modularity and assortativity. *Physical Review E*, 82(5):056113, 2010.

[112] Sergei Vassilvitskii and David Arthur. k-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1027–1035, 2006.

[113] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: is a correction for chance necessary? In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1073–1080, 2009.

[114] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *The Journal of Machine Learning Research*, 11:2837–2854, 2010.

[115] Stanley Wasserman, Katherine Faust, et al. *Social network analysis: Methods and applications*, volume 8. Cambridge University Press, 1994.

[116] Meng Wei, Wen Huang, Kyle A Gallivan, and Paul Van Dooren. Community detection by a Riemannian projected proximal gradient method. *IFAC-PapersOnLine*, 54(9):544–551, 2021.

[117] Guiyun Xiao, Zheng-Jian Bai, and Wai-Ki Ching. A columnwise update algorithm for sparse stochastic matrix factorization. *SIAM Journal on Matrix Analysis and Applications*, 43(4):1712–1735, 2022.

[118] X. Xiao, Y. Li, Z. Wen, and L. Zhang. A regularized semi-smooth Newton method with projection steps for composite convex programs. *Journal of Scientific Computing*, 76(1):364–389, Jul 2018.

[119] Xiantao Xiao, Yongfeng Li, Zaiwen Wen, and Liwei Zhang. A regularized semi-smooth newton method with projection steps for composite convex programs. *Journal of Scientific Computing*, 76(1):364–389, 2018.

[120] Liang Yang, Xiaochun Cao, Dongxiao He, Chuan Wang, Xiao Wang, and Weixiong Zhang. Modularity based community detection with deep learning. In *IJCAI*, volume 16, pages 2252–2258, 2016.

[121] Jieping Ye, Zheng Zhao, and Mingrui Wu. Discriminative k-means for clustering. *Advances in Neural Information Processing Systems*, 20:1649–1656, 2007.

[122] Hao Yin, Austin R Benson, Jure Leskovec, and David F Gleich. Local higher-order graph clustering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 555–564, 2017.

[123] Wayne W Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33(4):452–473, 1977.

[124] Thomas Zaslavsky. Signed graphs. *Discrete Applied Mathematics*, 4(1):47–74, 1982.

[125] Haijun Zhou. Distance, dissimilarity index, and network community structure. *Physical Review E*, 67(6):061901, 2003.

BIOGRAPHICAL SKETCH

Meng Wei was born in China. She earned her Bachelor degree in Applied Mathematics and Master degree in Statistics from Zhengzhou University in 2012 and 2015. She enrolled in the Ph.D. program at Florida State University in August 2015 and worked with Professor Kyle A. Gallivan, Professor Wen Huang and Professor Paul Van Dooren. In Spring 2022, she earned her second Master degree in Financial Mathematics from Florida State University. She currently works at Citibank as a Model Developer for counter-party credit risk models.