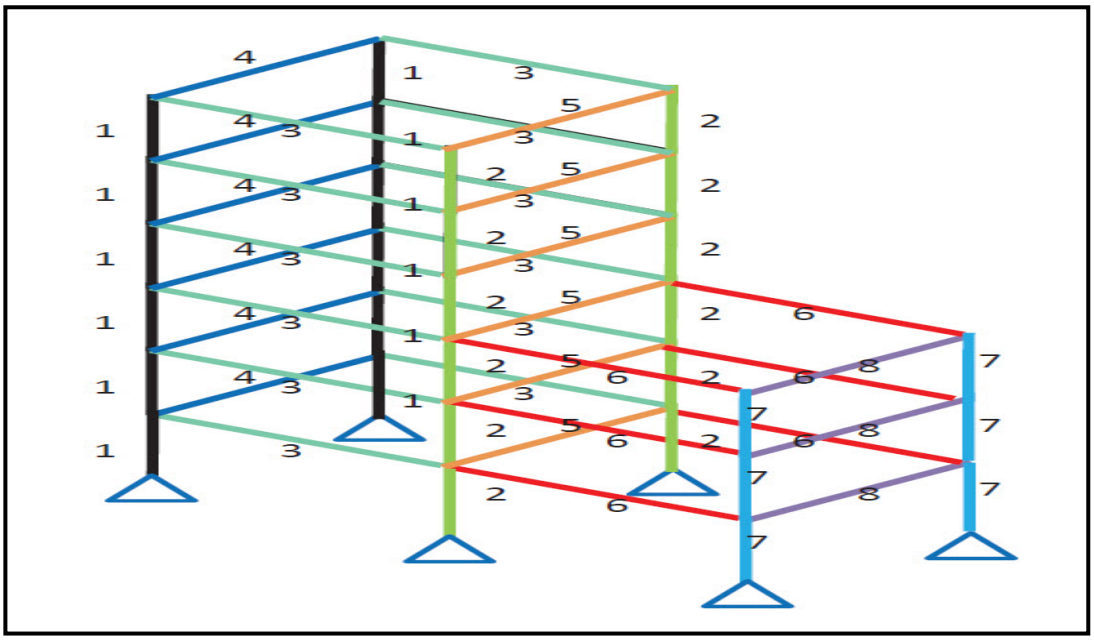
Categorical structural optimization : methods and applications

Huanhuan Gao

Par **Huanhuan GAO**

Categorical structural optimization: methods and applications

Thèse présentée en cotutelle pour l'obtention du grade de Docteur de l'UTC



Soutenue le 7 février 2019

Spécialité : Mécanique Numérique : Unité de recherche en Mécanique - Laboratoire Roberval (FRE UTC - CNRS 2012)

D2471

UNIVERSITÉ LIBRE DE BRUXELLES (ULB)
UNIVERSITÉ DE TECHNOLOGIE DE COMPIÈGNE (UTC)

DOCTORAL THESIS

Categorical Structural Optimization: Methods and Applications

Author:
Huanhuan GAO

Supervisor:
Dr. Piotr BREITKOPF
Prof. Thierry MASSART

Spécialité : Mécanique Numérique

7 février 2019

A thesis submitted in fulfillment of the requirements for the degree of Doctor

ULB-BATir and UTC-UMR CNRS 7337 (Roberval)
ULB-Building, Architecture, Town Planning and UTC-Mécanique

Declaration of Authorship

I, Huanhuan GAO, declare that this thesis titled, "Categorical Structural Optimization: Methods and Applications" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

"As heaven maintains vigor through movements, a gentle man should constantly strive for self-perfection. As earth's condition is receptive devotion, a gentle man should hold the outer world with broad mind. "

Confucius

UNIVERSITÉ LIBRE DE BRUXELLES (ULB) UNIVERSITÉ DE TECHNOLOGIE DE COMPIÈGNE (UTC)

Abstract

ULB-École Polytechnique de Bruxelles and UTC-Génie Mécanique, Acoustique et Matériaux
ULB-Building, Architecture, Town Planning and UTC-Mécanique

Doctor of Philosophy

Categorical Structural Optimization: Methods and Applications

by Huanhuan GAO

The thesis concentrates on a methodological research on categorical structural optimization by means of manifold learning. The main difficulty of handling the categorical optimization problems lies in the description of the categorical variables: they are presented in a category and do not have any orders. Thus the treatment of the design space is a key issue. In this thesis, the non-ordinal categorical variables are treated as multi-dimensional discrete variables, thus the dimensionality of corresponding design space becomes high. In order to reduce the dimensionality, the manifold learning techniques are introduced to find the intrinsic dimensionality and map the original design space to a reduced-order space.
The mechanisms of both linear and non-linear manifold learning techniques are firstly studied. Then numerical examples are tested to compare the performance of manifold learning techniques mentioned above. It is found that the PCA and MDS can only deal with linear or globally approximately linear cases. Isomap preserves the geodesic distances for non-linear manifold however, its time consuming is the most. LLE preserves the neighbour weights and can yield good results in a short time. KPCA works like a non-linear classifier and we proves why it cannot preserve distances or angles in some cases.
Based on the reduced-order representation obtained by Isomap, the graph-based evolutionary crossover and mutation operators are proposed to deal with categorical structural optimization problems, including the design of dome, six-story rigid frame and dame-like structures. The results show that the proposed graph-based evolutionary approach constructed on the reduced-order space performs more efficiently than traditional methods including simplex approach or evolutionary approach without reduced-order space.

In chapter 5, the LLE is applied to reduce the data dimensionality and a polynomial interpolation helps to construct the responding surface from lower dimensional representation to original data. Then the continuous search method of moving asymptotes is executed and yields a competitively good but inadmissible solution within only a few of iteration numbers. Then in the second stage, a discrete search strategy is proposed to find out better solutions based on a neighbour search. The ten-bar truss and dome structural design problems are tested to show the validity of the method. In the end, this method is compared to the Simulated Annealing algorithm and Covariance Matrix Adaptation Evolutionary Strategy, showing its better optimization efficiency.
In chapter 6, in order to deal with the case in which the categorical design instances are distributed on several manifolds, we propose a k-manifolds learning method based on the Weighted Principal Component Analysis. And the obtained manifolds are integrated in the lower dimensional design space. Then the method introduced in chapter 4 is applied to solve the ten-bar truss, the dome and the dame-like structural design problems.

Keywords: Categorical Optimization, Structural Optimization, Manifold Learning, Dimensionality Reduction, K-manifolds Learning, Evolutionary Methods, Kernel Function, Polynomial Fitting, Truss Structure, Weighted Principal Component Analysis

Acknowledgements

This research is supported by the China Scholarship Council (201406290021), the National Natural Science Foundation of China (No.11302173, No. 11620101002) and the Natural Science Foundation of Shaanxi Province (No. 2017JQ1037). I would like to express my sincere thanks to them.

I also want to thank my supervisor Prof. Rajan Filomeno Coelho. He has given so much care both on my research and my life. Even after he left ULB, he continued giving supervisions on my work. He is my first supervisor in Europe, I would never forget him.

I will give my gratitude to my supervisor Prof. Piotr Breitkopf. He has given me all kinds of help and supervision on my work, and has taught me a lot of research experience. Without him, I cannot finish my PhD career. Really thank you, Prof. Piotr Breitkopf.

I will also express my thanks to Prof. Thierry Massart has been in charge of all my administrative affairs in ULB since Prof. left ULB. He helped me with every registration in ULB and organized the yearly defenses and the final PhD defense for me. Thank you so much, Prof. Thierry Massart!

I also want to thank Prof. Manyu Xiao. Even though she worked in China, she also gave me a lot of help and advise on my scientific research. Wish her a fruitful research life.

I will also thank my father and two sisters. It is them who gave me the courage to complete my PhD career. Ten thousand miles far away, I cannot stop my missing them.

My dear friends, Dong Yang, Yingli Chen, Zhe Chen, Weide Huang, Liang Meng, Anna Madra, Jinhua Xiao, Shanggui Cai, Jing Wen, Congcong Ma and so on, gave me a lot of help with my study and my life. They also brought me plenty of happiness and delights. With them , I feel that friendship is everywhere.

Contents

List of Figures

List of Tables

List of Abbreviations

CSO	**C**ategorical **S**tructural **O**ptimization
DR	**D**imensionality **R**eduction
PCA	**P**rincipal **C**omponent **A**nalysis
KPCA	**K**ernel **P**rincipal **C**omponent **A**nalysis
WPCA	**W**eighted **P**rincipal **C**omponent **A**nalysis
MDS	**M**ulti-**D**imensional **S**caling
LLE	**L**ocally **L**inear **E**mbedding
RBF	**R**adial **B**asis **F**unction
GA	**G**enetic **A**lgorithms
EP	**E**volutionary **P**rogramming
ES	**E**volution **S**trategy
CMAES	**C**ovariance **M**atrix **A**daption **E**volution **S**trategy
SA	**S**imulated **A**nnealing
PSO	**P**article **S**warm **O**ptimization
ACO	**A**nt **C**olony **O**ptimization
SD	**S**teepest **D**escend
MMA	**M**ethod of **M**oving **A**symptotes
GMMA	**G**eneralized **M**ethod of **M**oving **A**symptotes
GCMMA	**G**lobally **C**onvergent **M**ethod of **M**oving **A**symptotes
MFD	**M**ethod of **F**easible **D**irections
SLP	**S**equential **L**inear **P**rogramming
SQP	**S**equential **Q**uadratic **P**rogramming
GRG	**G**eneralized **R**educed **G**radient

List of Symbols

$\mathbf{X}$	Coordinate matrix	
$\mathbf{Y}$	Coordinate matrix after dimensionality reduction	
$\mathbf{x}^j$	Coordinate of the j-th data point	
$\mathbf{y}^j$	Reduced ordered Coordinate of the j-th data point	
${}^k\mathrm{b}^j$	The k-th element of the coordinate of the j-th data point	
n	Number of data points	
M	Initial dimension of data points	
m	Reduced dimension of data points	
$\mathbf{C}$	Covariance matrix	
$\mathbf{\Phi}$	Eigenvector metric	
$\mathbf{\Lambda}$	Eigenvalue diagonal matrix	
Ψ	Kernel function matrix	
ζ	Kernel function	
$\mathbf{I}$	Identity matrix	
w_j	The j-th weight	
λ_i	The i-th eigenvalue	
$\boldsymbol{\phi}_i$	The i-th eigenvector	
ν	Poisson's ratio	
p	Polynomial order	
σ	Parameter of Gaussian function	
$\mathbf{K}$	Stiffness matrix	
$\mathbf{u}$	Displacement vector	
$\mathbf{P}$	Load vector	
$\mathbf{F}$	Internal force of a bar	
$\mathbf{F}_{cr}$	Critical load for linear local buckling	
$\mathbf{J}$	Design objective function	
$\mathbf{g}$	Design constraint functions	
θ	Angle variables	
$\mathscr{P}$	Path on the manifold	
H	Height	m
L	Length	m
t	Thickness	m
r	Radius	m
E	Young's Modulus	$\mathrm{N/m^2}$
G	Shear Modulus	$\mathrm{N/m^2}$
ρ	Density	$\mathrm{kg/m^3}$

Dedicated to my supervisors, friends and family...

Chapter 1

Thesis overview

In this chapter, an overview of the thesis is presented. In the first section, the objectives of the research are pointed out. Then we elaborate the general methodology of proposed optimization methods. In the last section, we give the outlines of the whole thesis.

1.1 Research objectives

The research objectives of the thesis are to improve the structural performances by using categorical optimization methods. More detailed objectives are listed as follows:

- to study the working mechanism of manifold learning methods and to compare their performance with numerical tests.
- to refine the nature of design variables of categorical optimization and to state the categorical structural optimization problems.
- to develop a discrete graph-based evolutionary approach and a continuous-discrete two stage strategy to deal with the structural optimization problems.
- to investigate how optimization algorithm parameters affect the optimization results and optimization efficiency.
- to compare the performance of the proposed methods with that of current mature optimization methods.
- to develop k-manifolds learning method to handle complex situations of the design space.
- to apply the proposed categorical optimization methods to truss structures composed with beams or bars.

1.2 Methodology

All the simulation processes in the thesis are carried out by using the finite element analysis packages of matlab software, including mass calculation, compliance evaluation and local linear buckling analysis. At first, the dimensionality of design space is reduced by certain manifold learning methods, then different methods are developed based on the reduced-order representations, including the graph-based evolutionary operators, the continuous-discrete two stage approach. In chapter 6, the design space of the optimization problems are firstly divided into several manifolds, and then dimensionality reduction is applied to each manifold. The proposed

optimization methods are tested on practical numerical structure design problems, showing their validity and optimization efficiency.

1.3 Thesis outline

In chapter 1, the thesis overview is given. The main goals of the research include the investigation of manifold learning methods, the statement of categorical optimization problems, the development and application of optimization methods and the evaluation of the proposed optimization methods. In the methodology part, the dimensionality of the design space is reduced by manifold learning methods, including the k-manifolds learning method. Then the discrete evolutionary approach and the two-stage search approach based on the reduced order representations are applied to structure design problems. The three principal parts, including manifold learning, simulation and optimization, are carried out with matlab packages. In the outline part, the general introduction of the whole thesis is presented.
In chapter 2, the research backgrounds and references are listed. Since the research content of the thesis is a combination of structural optimization and categorical optimization, both of them are detailed introduced, with different kinds of design criteria and optimization classification. In the second section, the related concepts of machine learning, together with their corresponding working mechanisms, are explained based on previous scientific works. Among all the machine learning methods, manifold learning are specially emphasized since it plays a key role in our methodology. The typical manifold learners, including both linear and non-linear ones, are elaborated with the aide of corresponding publications. The final section discusses the optimization methods which consist of metaheuristics and gradient-based methods, together with comments on their performance.
Chapter 3 mainly introduces the mechanisms of several manifold learning techniques, including principal component analysis (PCA), multi-dimensional scaling (MDS), Isomap, locally linear embedding (LLE) and kernel principal component analysis (KPCA). Then three testing examples are used to compare the manifold learning results. Besides, the learning efficiency is also discussed. As a focusing point, KPCA is specially discussed on its failure to length and angle preservation.
Chapter 4 presents a discrete evolutionary graph-based approach to handle categorical optimization problems. This approach deals with multi-objective optimization with a single manifold design space. The higher dimensional design space is firstly mapped to lower dimensional pace by applying the non-linear leaner Isomap, then the design instances are connected using k-means or r-means. The evolutionary optimization operators including crossover and mutation are developed based on the lower dimensional connecting graph. Finally, the algorithm is tested by numerical examples including the dome and dam structure and six-story rigid frame structure. The optimization results and efficiency are compared with a continuous method and simplex approach, showing the advantages of the proposed algorithm.
In chapter 5, a two-stage search approach is proposed to handle single-objective optimization problems with a single manifold design space. In the first stage, the original design space is mapped to a lower dimensional space by LLE, then we construct the analytical mapping from lower dimensional space to original one. Thus the design problem is transformed to a lower order one. We make the discrete problem continuous and utilize the gradient-based method to execute a rough search. In the second stage, a neighbour search based on the connecting graph is developed and applied to numerical examples. The proposed approach is also compared with

metaheuristics. In order to improve the optimization efficiency, a gradient-based fast neighbour search is proposed for the second search stage.
Chapter 6 proposes a k-manifolds learning methods aiming at finding multiple manifolds in the observation space and its application to single-objective categorical structural optimization problems. Before dimensionality reduction, each design instance is attached with a weight which donates the contribution in the process of PCA. By optimizing the weights, the k-manifolds are separated automatically. Then the lower dimensional representations of k-manifolds are organized and merged in the same space. The categorical structural optimization is executed based on the lower dimensional space of k-manifolds. Numerical examples are also tested in order to show the validity of the methods.
Chapter 7 presents the conclusions and perspectives of the whole thesis. The conclusions include the evaluations of different kinds of manifold learning methods, the effectiveness and optimization efficiency of three categorical optimization methods: the discrete evolutionary approach, the two-stage search methods and k-manifolds learning based approach. The further development and perspectives of the proposed methods are also given in this chapter.

Chapter 2

Research background

2.1 Categorical structural optimization

As a powerful design and selection tool, optimization is widely used or embedded in both research subjects and engineering design applications due to its strong mathematical background, flexible interfaces, various forms and design automation property, for example, the identification and characterization problems (also named the inverse problems) (Mosselman, Polman, and Dijkema, 1996), machine learning subjects (Tenenbaum, De Silva, and Langford, 2000), route planning (Fu, Ding, and Zhou, 2012), structural optimization (Cheng and Guo, 1997; Bendsøe, 1989) et al.
Among all the applications of optimization techniques, structural optimization is "a discipline dealing with optimal design of load-carrying mechanical structures" (by Krister Svanberg). The general framework of structural optimization includes an optimizer part and a simulation part (or response estimation). The role of the optimizer is to update the designs iteration by iteration according to performance of the structure. In gradient-based optimizer, the sensitivities of all the criteria with respect to each design variables are requested to renew the designs. While in population-based evolutionary optimizers, the sensitivities are not necessary, and the population is updated in successive generations.
The main task of the simulation (Fig. 2.1) part is to calculate the responses of the structure, including weight (mass) or volume (Kaufmann, Zenkert, and Wennhage, 2010), structural compliance (or strain energy) (Sigmund, 2001), maximum stress (Duysinx and Bendsøe, 1998), natural vibration frequency (Kaveh and Zolghadr, 2011), dynamic responses (Biegler, 1984), local or global buckling (Lund, 2009) et al. The loads which are applied to the structure contain static loads (Sigmund, 2001), acceleration loads (including gravity) (Xie and Steven, 1997), time-variant dynamic incentives (Branke et al., 2000), thermal loads (Li et al., 1999), electromagnetic loads (Chun et al., 1997) et al.
From the formal point of view, optimization may be classified by different kinds of features. According the number of objectives, we can divide optimization problems into single objective optimization (Liang et al., 2014) and multi-objective optimization (Deb et al., 2002). Furthermore, considering the number of design constraints, we can classify optimization into constrained problems (Powell, 1978) and unconstrained problems (Dennis Jr and Schnabel, 1996). One can also classify optimization types by methods used, for example, the gradient-based optimization (Snyman, 2005), the evolutionary optimization (Davis, 1991), memetic optimization (Moscato, 1989) et al.
The so-called categorical optimization is labelled from the optimization classification based on the nature of design variables. In the principal level, we can divide optimization problems (Fig. 2.2) into continuous optimization, non-continuous optimization and mixed-variable optimization. Among them, continuous optimization

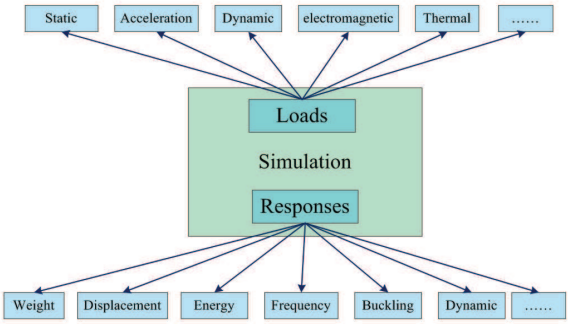


FIGURE 2.1: The loads and responses of simulation

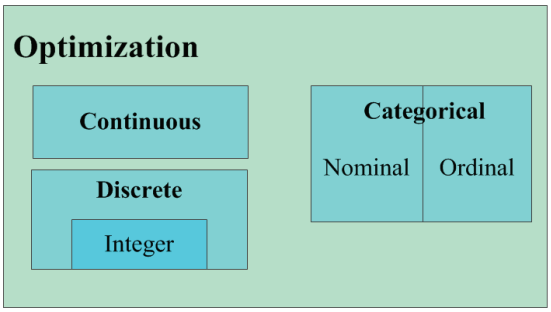
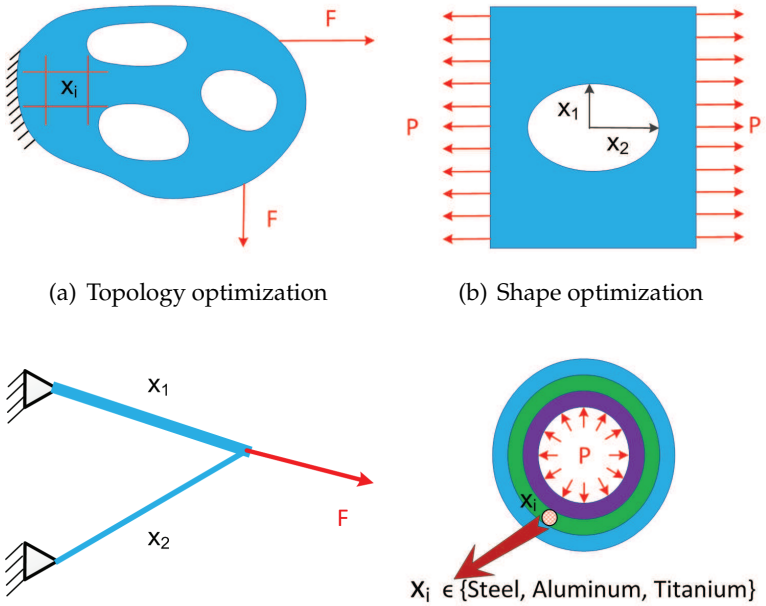


FIGURE 2.2: Classifications of optimization by the nature of design variables

denotes the optimization problems in which the design variables are all continuous (Pan et al., 2010; Zhao et al., 2011). While non-continuous optimization includes discrete optimization, categorical optimization. The discrete optimization (Deo and Kowalik, 1983; Rajeev and Krishnamoorthy, 1992) indicates that the design variables are all discrete (or integer), which means the corresponding design space collapses into a set of discrete points. We need to note that, in discrete optimization, the design variable can only take scalar values, despite of that it may be handled with binary strings or other means. Different from the discrete one, categorical optimization (Coelho et al., 2015; Sloane and Morgan, 1996) can be defined as the optimization where design variables (we say categorical variables) can take the values of a finite set of instances, which were regarded as non-numerical ones in the beginning. While in our work, we represent those instances with vectorial data, which can clearly reveal the nature of categorical variables: they are multi-dimensional discrete variables. And this point will be elaborated in more detail.

From the viewpoint of problem types, structural optimization can be subdivided into topology (Sigmund, 2001), shape (Rozvany, Zhou, and Birker, 1992), size (Laporta and Brussard, 1991), material selection (Stegmann and Lund, 2005), cross-section selection et al (Fig. 2.3). Among these categories, the topology optimization (Fig. 2.3(a)) deals with the layout design of materials or parts under given set of loads and boundary conditions, for example the topology optimization of continuum structure (Duysinx and Bendsøe, 1998) and the truss topology optimization (Rozvany, 1996). The former is usually treated with continuous variables, meanwhile the latter is always handled in a discrete manner. The shape optimization (Fig. 2.3(b)) optimizes the boundary of a part, for example the radius of structural



(a) Topology optimization (b) Shape optimization

(c) Size optimization (d) Material selection

FIGURE 2.3

holes (Lee and Kim, 2010a). The widely used shape optimization techniques include geometrical parameter control (Hilbert et al., 2006), Isogeometry (Wall, Frenzel, and Cyron, 2008), level-set functions (Allaire, Jouve, and Toader, 2002), control points (Nemec, Zingg, and Pulliam, 2004) et al. Size optimization (Fig. 2.3(c)) (Kaveh and Talatahari, 2009) focuses on the geometrical parameters of a structural feature, for example the length of a bar, the thickness of a shell. The material selection (Fig. 2.3(d)) aims at finding the best material combinations for different parts of the structure (Zhou, Yin, and Hu, 2009; Ashby, 2000). To handle the material selection problems, both continuous methods and discrete methods can be applied.

The categorical structural optimization is the combination of categorical optimization and structural optimization, for instance, the selection of a material within a finite the set (steel, aluminium, titanium, composites) or the selection of cross-section types among the catalog (square, I-shaped, round,...). The key issue with this kind of problems is how to represent the design variables (Coelho et al., 2015; Filomeno Coelho, 2014).In previous work, different coding schemes including binary, real and simplex, are proposed successively. Binary coding (Goldberg, 2006) can only represent non-continuous values with 0/1 strings which may lead to the problems of expression accuracy, excess non-sense strings et al. Real coding utilizes the real values to stand for representation and optimization, thus the genotype and phenotype are united into one (Herrera, Lozano, and Verdegay, 1998; Eshelman and Schaffer, 1993). But when dealing the categorical variables, both binary coding and real coding will casually assign unfounded distances (or called relative positions) to categorical data. To overcome the distance expression difficulty, the simplex coding (Filomeno Coelho, 2014; Herrera et al., 2014) is proposed and applied in categorical optimization. Instead of using binary strings or real numbers, simplex coding exploits vectorial coordinates of points in the design space, in which different categorical design candidates are assigned as the same. The other optimization methods which are tailored for categorical optimization include the mesh adaptive direct

search (Abramson et al., 2009), ant colony (Liao et al., 2014), mixed variable programming (Kokkolaras, Audet, and Dennis, 2001) and global descent approach (Lindroth and Patriksson, 2011).

2.2 Machine learning and manifold learning

The objective of manifold learning (Lunga et al., 2014; Tenenbaum, 1998) is to find the potential lower dimensional topological space embedded in the higher dimensional space. The original higher dimensional space is named observation space while the lower dimensional space is called feature space. Manifold learning is a kind of unsupervised machine learning method. The various machine learning theories and methods can be generally divided into three learning method groups: distinction construction and classification, conditional probability construction and probability density function construction through regenerative models.
The distinction construction and classification methods include clustering analysis, pattern recognition, artificial neural networks, support vector machines, manifold learning, Bayesian classifier and decision tree.
Among those learning methods, clustering analysis (Hartigan, 1975; Jain and Dubes, 1988; Hartigan and Wong, 1979) groups a certain number of sample points in different clusters based on the distance definition (for example the Euclidean distance) or on the similarity according to data features, and it has been applied in economics, computer science and web techniques.Pattern recognition (Nasrabadi, 2007; Anzai, 2012) concentrates on finding of data regularities and patterns, and assigns different labels to input values according to the learning results. Inspired by the biological neural networks, artificial neural networks (Schalkoff, 1997; Yegnanarayana, 2009) exploit the multi-layer neuron which work as simple linear or non-linear classifiers to distinguish different kinds of categories through a big amount of learning examples. The artificial neural networks also have a wide application in artificial intelligence, vehicle control, face identification, signal classification. They are also referred as a king of deep learning. Support vector machines (Suykens and Vandewalle, 1999; Furey et al., 2000) are a kind of supervised learning methods which can analyse the training data and build up learning models for classification and regression analysis. Manifold learning (Lunga et al., 2014; Tenenbaum, 1998) aims to find the lower dimensional representation of higher dimensional data with no (or less) information loss. It is also regarded as a kind of dimensionality reduction technique. It is used in image management, structural optimization. We will give a detailed introduction to manifold learning later in chapter 2. The Bayesian classifiers (Wang et al., 2007; Domingos and Pazzani, 1997) are sets of probabilistic classifiers taking advantages of the Bayes' theorem by assuming that the features are strongly independent (naive Bayes). Bayesian classifiers have achieve a great success in text check, speech recognition and human-level concept learning (Lake, Salakhutdinov, and Tenenbaum, 2015). The decision tree (Quinlan, 1986; Quinlan, 1987) applied the tree model with decisions and their possible consequences in order to make decisions. It is simple and well interpreted, but it may also bring inaccuracy in result outputs , thus cause wrong decisions.
The second group of machine learning methods: conditional probability construction, mainly contain the Gaussian process regression, linear discriminant analysis, nearest neighbor methods, radial basis function kernels (Chang et al., 2010; Vert, Tsuda, and Schölkopf, 2004). In Gaussian process regression, the lazy learning and

the kernel function are used to predict the value of given training data, providing both of the value estimation and the uncertainty information. The Gaussian process learning can be a helpful tool to predict and build up the surrogate model of a function, for example the kriging model. Linear discriminant analysis (Mika et al., 1999; Altman, Marco, and Varetto, 1994; Prince and Elder, 2007) applies a linear combination of features to classify or separate different kinds of events, and it is commonly used in pattern recognition, statistics and machine learning. The nearest neighbor methods (Shakhnarovich, Darrell, and Indyk, 2006; Devroye and Wagner, 1982) can be regarded as a kind of lazy learning. In the nearest neighbor methods, the investigated function is approximated locally, then the object is classified only by a majority vote of its neighbors. Due to its simplicity, the nearest neighbor methods are widely used in classification and regression fields. The radial basis function kernels (Chang et al., 2010; Vert, Tsuda, and Schölkopf, 2004) are also named Gaussian RBF. They are popular in all the kernel-related learning methods, for example the support vector machine and the kernel principal component analysis. And in chapter 2, we give an exhaustive elaboration of the radial basis function.
The third learning algorithm group is the probability density function construction through regenerative models. They can be divided to expectation-maximization algorithm, probability models and generative topographic mapping. Among them, the expectation-maximization algorithm (Zhang, Brady, and Smith, 2001; Moon, 1996; Fessler and Hero, 1994) deals with the statistical models which are dependent on unobserved latent variables. The method can find the maximum likelihood estimates of the parameters in an iterative manner. The probability models (Barlow and Proschan, 1975; Ross, 2014) can build the relationship between several independent random variables by using the graph theory. The graphs used in probability models include the Bayesian networks and the Markov networks. This method is usually used in Bayesian learning and machine learning. The generative topographic mapping (Bishop, Svensén, and Williams, 1998) projects a low dimensional data to a high dimensional space by a smooth function considering noise, then the expectation-maximization algorithm is used to learn the parameters of the low dimensional distribution and the noise. The advantage of this method is that the shrinking neighbourhoods or decreasing step lengths are not necessary for algorithm convergence.
There are two kinds of linear manifold learning methods which are named Principal Component Analysis (PCA) (Jolliffe, 1986; Wold, Esbensen, and Geladi, 1987) and Multi-Dimensional Scaling (MDS) (Martin and Eroglu, 1993; Kandogan, 2001), respectively. PCA investigates the covariance matrix of the observed data and decomposes it through eigenvalue decomposition. Consequently, PCA reorganizes the projection system by minimizing the variance, thus it can preserve the most of the information. MDS acts on the inter-distances between data points. It focuses on the preservation of Euclidean distances. Normally, the results obtained by PCA and MDS are the same, but PCA provides an explicit expression which can map new extra points while MDS cannot handle that. The main disadvantage of PCA and MDS is that they are not able to handle non-linear data, for example the "Swiss Roll" manifold (Tenenbaum, De Silva, and Langford, 2000).
The well-known non-linear manifold learning methods include Locally linear Embedding (LLE), Isomap and Kernel Principal Component Analysis (KPCA). LLE (Roweis and Saul, 2000; De Ridder et al., 2003; Donoho and Grimes, 2003) consists of three steps: Firstly, LLE builds the neighbourhood for each sample point by using either k-neighbour means or r-radius means; Secondly, LLE constructs the weights of all neighbours by minimizing the construction error. One of the advantages is that the weight optimization problem is convex and thus local minima can be avoided.

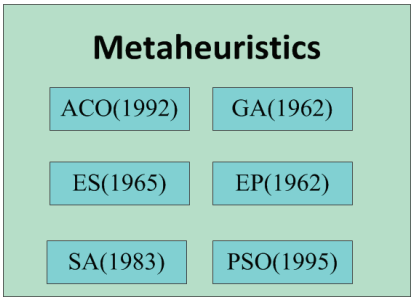


FIGURE 2.4: Methods of Metaheuristics and corresponding publication years

Isomap (Tenenbaum, De Silva, and Langford, 2000; Balasubramanian and Schwartz, 2002) can be regarded as a non-linear version of MDS. What is different between MDS and Isomap is that Isomap applies the geodesic distances instead of Euclidean distances. In Isomap, the geodesic distance is calculated by the Dijkstra algorithm (Dijkstra, 1959), then the distances are preserved in the lower dimensional design space. The dimensionality reduction effects of Isomap actually heavily depends on the construction of the neighbourhood connecting graph. If the graph cannot reflect the true manifold in the higher dimensional space, the Isomap will fail. Also, Isomap has the same disadvantages as MDS, namely Isomap cannot deal with new data points and needs the target dimension number is given in advance. KPCA (Schölkopf, Smola, and Müller, 1998; Cao et al., 2003) firstly maps the original data to much higher dimensional space by using the kernel functions, for example the linear kernel function, the polynomial kernel function and the Gaussian radial basis function. Then PCA is utilized to reduce the dimensionality. This is the classical idea of Support Vector Machines (Cortes and Vapnik, 1995; Vapnik, 2013). The dimensionality reduction effects of KPCA mainly depend on the choice of kernel functions and the parameters of the kernel functions. Those non-linear manifold learning techniques have already been used in identification problems (Meng et al., 2015), structural optimization (Raghavan et al., 2013), visualization problems (Patwari, Hero III, and Pacholski, 2005; Wang et al., 2008) and networks (Patwari and Hero, 2004).

2.3 Optimization methods

2.3.1 Metaheuristics

The metaheuristics denote a process of generating and selecting good solutions within a limited acceptable time cost. Normally the global optima cannot be guaranteed or proved by metaheuristics. Despite of the cons, metaheuristics still have received more and more attentions and are applied to handle numerous engineering and research optimization problems.
The popular representative methods of metaheuristics (Fig. 2.4) include ant colony optimization (ACO), genetic algorithms (GA), evolution strategy (ES), evolutionary programming (EP), simulated annealing (SA) and particle swarm optimization (PSO).
The ant colonies optimization (Dorigo and Birattari, 2011; Dorigo, Maniezzo, and

Colorni, 1996) was firstly used to find the optimal path based on the food-seeking behaviours of ants. Then it is extended and diversified to solve many kinds of optimization problems. It is a kind of simulated evolutionary algorithm. Nowadays, the ant colonies optimization is usually used in route planning, scheduling, resource allocation problems et al. The genetic algorithms (Davis, 1991; Anderson-Cook, 2005) are a kind of nature-inspired method to deal with optimization problems. They simulate the evolution of biological populations and apply the "survival of the fittest" idea. The main operators of genetic algorithms consists of initialization, crossover, mutation and selection. Due to the flexibility of coding schemes, operator extensibility and interface versatility, genetic algorithms have achieved much success both in theoretical and engineering applications (Goldberg and Holland, 1988; Srinivas and Deb, 1994). In evolution strategy (Janis, 1976), the design evolution is inspired only by mutation and selection in one generation. The evolution continues until the termination conditions are satisfied. In order to generate new populations, the normal distribution is used to generate new random individuals. The mutation strength is controlled by self-adaptation schemes. If the individual step size is determined by covariance matrix, the corresponding algorithm is named covariance matrix adaptation evolution strategy (CMA-ES) (Hansen, Müller, and Koumoutsakos, 2003; Ros and Hansen, 2008). evolutionary programming (Back, 1996) has many similarities with genetic programming. In EP, the new variation is only driven by mutation operators which act on the population, then the survivor selection is applied to maintain good individuals for next generation. The applications of EP include artificial intelligence (Fogel, Owens, and Walsh, 1966; Fogel, 1999), machine learning, structural optimization. The simulated annealing (Kirkpatrick, Gelatt, and Vecchi, 1983) is a probabilistic optimization method for finding the global optimum of the objective over a large search space within a fixed time cost by simulating the process of metal annealing. The whole simulation process is controlled by the annealing temperature, which determines by how much probability the new design is accepted or rejected. By accepting and rejecting operation, the design has a certain possibility to walk out of the current local optima and find the global one. The particle swarm optimization (Kennedy, 2011; Poli, Kennedy, and Blackwell, 2007) is also a famous population based optimization method. The method regards every individual as a micro particle without any volume and mass, and every particle moves in the design domain with a certain speed according to the flying experiences of its own and its neighbours. All the particles are supposed to concentrate on better solutions. In further development, the inertia weights are introduced to control the exploitation and exploration properties. When orthogonal learning (Zhan et al., 2011; Ho et al., 2008) is introduced into PSO, the global convergence, convergence accuracy and robustness have been significantly improved.
Other metaheuristics optimization methods include artificial immune systems (AIS) (Aickelin, Dasgupta, and Gu, 2014), cultural algorithms (CA) (Reynolds, 1994), coevolutionary algorithms (CEA) (Michalewicz and Nazhiyath, 1995), differential evolution (DE) (Storn and Price, 1997), genetic programming (GP) (Koza, 1992), greedy adaptive search procedures (GASP) (Feo and Resende, 1995).

2.3.2 Gradient-based methods

Different from metaheuristics, the gradient-based methods also represent a large group of iterative design updating methods taking the advantage of gradients (also called sensitivities) of the objective function and constraint functions with respect

to design variables. For multiple variable optimization problems, the first order sensitivities can be organized in the form of Jacobian vector, and the second order sensitivities are written in the form of Hessian matrix.
The basic gradient-based methods include the steepest descent (SD), conjugate gradient (CG), Newton's method (NM), as shown in Fig. 2.5. The steepest descent (Goldstein, 1965; Yamada, 2001) is a classical first-order optimization method. It utilizes the reverse direction of the gradients as the searching direction, and finally one can expect that the searching process converges in a local solution. The advantages of steepest descent is that it is simple, but it suffers from the so called "sawtooth phenomenon" which leads to a slow convergence. Furthermore, the steepest descent method does not consider design constraints, so modifications are necessary, for example the penalty function methods (Kort and Bertsekas, 1972). The conjugate gradient (Powell, 1977) is also method to solve unconstrained optimization problems. By applying the conjugate property of the adjacent searching directions in the optimization loops, the conjugate gradient introduces a modification of current searching direction, making it faster than the steepest descent in the beginning, but this modification may also be not conducive to the convergence speed due to error accumulation in final several iteration. The Newton's method (Wedderburn, 1974; Loke and Barker, 1996) (also named Newton-Raphson method) uses both of the first-order and second-order gradients (Hessian matrix) to construct a second-order approximation of the objective function. Many analytical tests have proved that in the case where the Newton's method is applicable, it is faster than the steepest descent. However, one of the difficulties for Newton's method is that the Hessian matrix is usually hard to obtain. And an fatal condition required is that the Hessian matrix must be positive definite. In order to solve this problem, a series of quasi Newton's methods (Shanno, 1970) are proposed and developed, including the DFP algorithm (Broyden, 1970) and BFGS algorithm (Zhu et al., 1997). The idea of both algorithms is to approximate the Hessian matrix in order to complete optimization search under the quasi Newton's conditions.
The practical gradient-based optimization methods (Fig. 2.5) used for engineering problems include the method of feasible directions (MFD), sequential linear programming (SLP), sequential quadratic programming (SQP), generalized reduced gradient (GRG), method of moving asymptotes (MMA) et al. The method of feasible directions (Zoutendijk, 1960; Chen and Kostreva, 1999) can be regarded as an extension of unconstrained gradient descent. It starts at the initial design point, and applies the 1-D line search along the descent direction. The key step is the choice of search directions and searching step lengths. The method of feasible directions can solve the optimization problems with inequality constraints at a fast speed, however, it can not deal with the optimization problems with equality constraints. Sequential linear programming (Marcotte and Dussault, 1989; Etman et al., 1996) is proposed to handle non-linear optimization problems by using first-order approximation. This method solves a sequence of first-order approximations of the original problem, which provides a high optimization efficiency. At last, the trust regions techniques are introduced to guarantee converged optimization process. Similar as the sequential linear programming, the sequential quadratic programming (Nocedal and Wright, 2006; Boggs and Tolle, 1995) solves a sequence of sub-problems which are approximated by quadratic models with linear constraints. If the optimization problem is unconstrained, the sequential quadratic programming degenerates to Newton's method; If only equality constraints exist, this method evolves to Newton's method with Karush–Kuhn–Tucker conditions of the optimization problem. The generalized reduced gradient (Lasdon et al., 1978) is the general

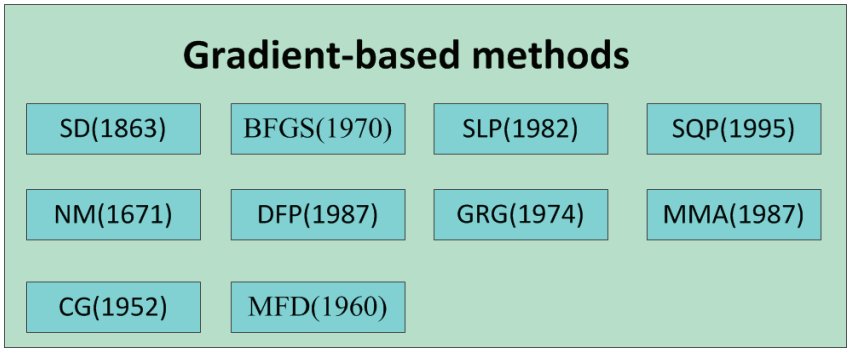


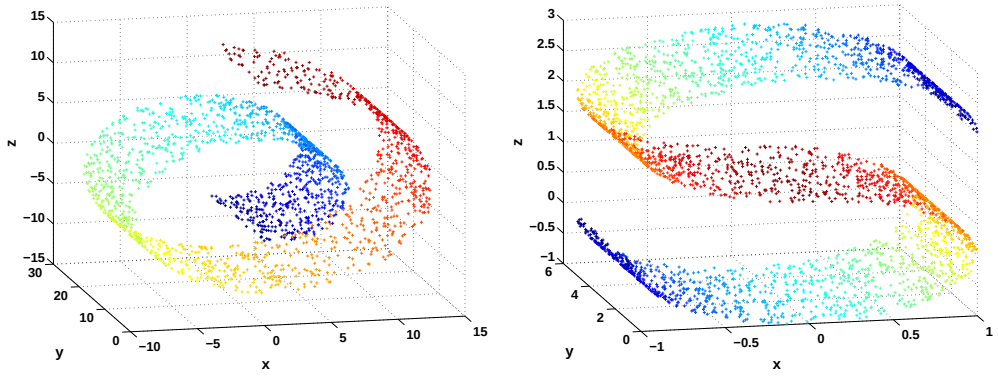
FIGURE 2.5: Gradient-based methods and the publication years

form of the reduced gradient method (Wolfe, 1962; Minyi and Jiye, 1979) to solve optimization problems together with non-linear objective function and non-linear constraints. In this method, the inequality constraints are firstly transformed into equality constraints by introducing a positive slack variable. Then the all the design variables are divided into independent and dependent variables according to the all the equality constraints, consequently the dependent variables are represented by the independent ones. After that, the size of the gradient information is reduced. Then the Newton's method is applied to solve the non-linear equation sets and obtain a proper searching step length until the algorithm is converged. The method of moving asymptotes (Svanberg, 1993; Svanberg, 1987) uses the asymptotes iteration by iteration to control the constructed convex sub-problems being solved, which can help improve the convergence speed and optimization stability. The further developments include the generalized method of moving asymptotes (Zhang et al., 1996) and the globally convergent method of moving asymptotes (Zillober, 1993).

Chapter 3

Manifold learning techniques

In this chapter, we give an introduction of several popular manifold learning algorithms. The role of manifold learning is crucial because all the proposed optimization methods are developed based on the manifold learning techniques. All the manifold learning methods assume that the data points are randomly sampled from an m-dimensional manifold which exists potentially in an M-dimensional space (m<M). The task of manifold learning is to find the new representations (coordinates) of those sample points in the lower m-dimensional space, according to their coordinates in the higher M-dimensional space. Conventionally, the original higher dimensional space is called observation space, while the new lower dimensional space is named feature space.
The manifold learning algorithms are mainly categorized into two kinds: the linear and non-linear ones. There have already been theories and applications which concentrate on the linear manifold learning, for example, a linear manifold indicates a 1-D line, 2-D plane, or a hyperplane in the higher dimensional observation space. Although in some cases, the potential manifold is assumed to be linear, it is generally non-linear in most practical cases. Another fact is that since the data points are obtained by measurements or observations, it is unavoidable to introduce the observation error, or we say noise, which will undermine the original linearity of the data. Consequently the linear manifold is always used as a realistic assumption rather than an ideal tool. Even though, we cannot ignore the successful theory developments and numerous practical applications of linear manifold learning techniques because firstly non-linear manifold can always be regarded as locally linear and secondly, linear manifold learning techniques usually have high efficiency in dealing with non-linear data, especially for big data and large dimensionality numbers.
The non-linear manifold learning takes more complex manifolds in consideration, for example, manifolds with bending, warping or twisting. The famous examples for scientific research purposes are the "Swiss Roll" and the "S-Curve", as shown in Fig. 3.4. Undoubtedly, non-linear manifolds are closer to the real data status of the world. The task of non-linear manifold learning is to recover the lower dimensional coordinates with provided higher dimensional representations of the non-linear data.
The well-known linear manifold learning algorithms include Principal Component Analysis (PCA) and Multidimensional Scaling (MDS) illustrated in section 3.1 and 3.2. Three typical non-linear manifold learning techniques are also introduced in this chapter, including Isomap (section 3.3), Locally Linear Embedding (LLE) (section 3.4) and Kernel Principal Component Analysis (KPCA) (section 3.5), respectively. In section 3.6, we give testing examples of the proposed manifold learning techniques.



(a) Swiss roll (b) S-Curve

FIGURE 3.1: Examples of non-linear manifolds

3.1 Principal component analysis

We firstly define the data $\mathbf{X} = [\mathbf{x}^1, \mathbf{x}^2, \cdots, \mathbf{x}^n]$ is a M $\times$ n matrix. M is the number of dimensions, while n is the number of data points. Dimensionality reduction is to find a lower m-d representation of the n data points.
Suppose the initial data $\mathbf{X}$ is centered in the beginning, then the covariance matrix can be defined as:

$$\mathbf{C} = \frac{1}{n}\mathbf{X}\mathbf{X}^T. \tag{3.1}$$

The eigenvalue decomposition yields:

$$\mathbf{C} = \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^T, \tag{3.2}$$

where $\mathbf{\Lambda}$ denotes the eigenvalues and $\mathbf{\Phi}$ is the metric of corresponding eigenvectors, and $\mathbf{\Phi}^T\mathbf{\Phi} = \mathbf{I}$. Note that after a descending ordering

$$\begin{aligned} &\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \lambda_2, \cdots, \lambda_M), \\ &\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_M. \end{aligned} \tag{3.3}$$

We choose the first m eigenvectors and form the projection matrix $\mathbf{\Phi}_m = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \cdots, \boldsymbol{\phi}_m]$. Finally, we obtain the representation in m dimensional space:

$$\mathbf{y}^i = \mathbf{\Phi}_m^T \mathbf{x}^i, \;\; i = 1, 2, \cdots, n. \tag{3.4}$$

With Eq. 3.4, we can project new points to the lower dimensional space.
After dimensionality reduction, the data information preservation ratio of the lower dimensional representation can be measured by:

$$\kappa = \frac{\sum_{i=1}^{m} \lambda_i}{\sum_{i=1}^{M} \lambda_i}. \tag{3.5}$$

$\kappa = 1$ means no information loss during the mapping process and the m is the number of intrinsic dimensionality.

3.2 Multidimensional scaling

Multidimensional scaling is also a linear manifold learning algorithm. Different from PCA, MDS focuses on the distance preservation. Given the distance matrix:

$$\begin{aligned} &\mathbf{D}^{(\mathbf{X})}_{n\times n}, \mathbf{D}^{(\mathbf{X})}(i,j) = ||(\mathbf{x}^i, \mathbf{x}^j)||_2, \\ &i,j = 1,2,\cdots,n, \end{aligned} \tag{3.6}$$

where $||\cdot||_2$ is the 2-norm of a vector, MDS is developed to find the new coordinates $\mathbf{Y}$ in lower dimensional space:

$$\begin{aligned} &\mathbf{Y}_{m\times n} = [\mathbf{y}^1, \mathbf{y}^2, \cdots, \mathbf{y}^n], \\ &\mathbf{y}^i = ({}^1b^i, {}^2b^i, \cdots, {}^mb^i)^T, \ \ i = 1,2,\cdots,n, \\ &\sum_{i=1}^{n} {}^lb^i = 0, \ \ l = 1,2,\cdots,m. \end{aligned} \tag{3.7}$$

In order to handle centred data, the transformation matrix $\mathbf{P}$ is defined as:

$$\mathbf{P} = \mathbf{I} - \frac{1}{n}\mathbf{e}\mathbf{e}^T, \mathbf{e}^T = (1,1,1,\cdots,1)_{1\times n}, \tag{3.8}$$

yielding the Gram matrix $\mathbf{G}$:

$$\mathbf{G} = -\frac{1}{2}\mathbf{P}(\mathbf{D}^{(\mathbf{X})})^2\mathbf{P}^T \tag{3.9}$$

Because $\mathbf{G}$ is symmetric and positive definite, we carry out the eigenvalue decomposition and get:

$$\mathbf{G} = \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^T, \tag{3.10}$$

where

$$\begin{aligned} &\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \lambda_2, \cdots, \lambda_M), \\ &\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_M \end{aligned} \tag{3.11}$$

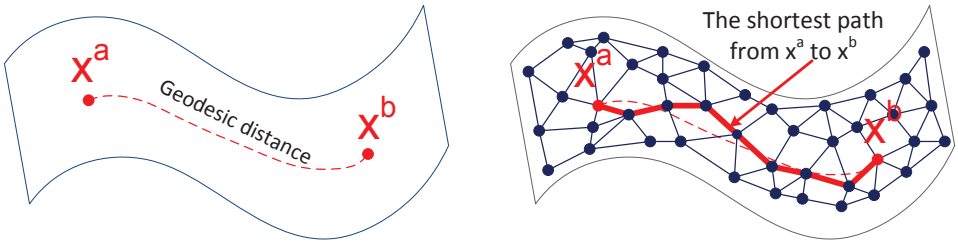
and $\mathbf{\Phi}$ is the matrix whose columns contain eigenvectors of $\mathbf{G}$ and λ_i corresponds to the i-th largest eigenvalue.
We choose the m largest eigenvalues and define:

$$\mathbf{\Lambda}_m^{\frac{1}{2}} = [\mathrm{diag}(\sqrt{\lambda_1}, \sqrt{\lambda_2}, \cdots, \sqrt{\lambda_m})] \tag{3.12}$$

Finally we obtain the reduced-order coordinate matrix $\mathbf{Y}$:

$$\mathbf{Y} = [\mathbf{y}^1, \mathbf{y}^2, \cdots, \mathbf{y}^n] = \mathbf{\Lambda}_m^{\frac{1}{2}}\mathbf{\Phi}_m^T. \tag{3.13}$$

where $\mathbf{\Phi}_m$ is composed by the eigenvectors corresponding to the m largest eigenvalues.
Different with PCA, MDS cannot add new points after the the process is completed. Another difference is that MDS needs the target dimensionality number given in advance, while PCA can decide the target dimensionality number after eigenvalue decomposition.



(a) Geodesic distance (b) The shortest path

FIGURE 3.2: Illustration of the geodesic distance

3.3 Isomap

As a non-linear manifold learning method, Isomap can be regarded as a variant of MDS. The difference between MDS and Isomap is that MDS uses the Euclidean distances while Isomap utilizes the geodesic distances. Geodesic is the line along which the acceleration vector equals zeros on the manifold. In Isomap, the geodesic distance is approximated by the shortest path based on graph algorithm.
Fig. 3.2(a) shows the illustration of a geodesic from point $\mathbf{x}^a$ to $\mathbf{x}^b$. In order to calculate the geodesic distance, we first define the nearest neighbour points for all the points by the ε-mean or the K-mean algorithm(Tenenbaum, De Silva, and Langford, 2000). Then, for every pair of neighbour points, their geodesic distance is approximated by the Euclidean distance; for other nonadjacent points, their geodesic distance is approximated by summing up the distances along the shortest path obtained by Dijkstra algorithm (Dijkstra, 1959) in the graph (Fig. 3.2(b)).
After obtaining the distance matrix, the MDS algorithm is run and yields the new representation in the reduced-order space.

3.4 Locally linear embedding

The thought of Locally Linear Embedding (LLE) is that a manifold in a small local neighbourhood can be regarded as a linear one. Thus, a point can be represented by the linear combination of the coordinates of its neighbour points. The combination coefficients are actually the description of the local environment of the point. And this description, namely the coefficients, are preserved in the new reduced-order space.
The first step of LLE is to identify K neighbours per data point with the ε-mean or the K-mean. The reconstruction errors are calculated by the cost function:

$$E_1(W) = \sum_i |\mathbf{x}^i - \sum_i W_{ij}\mathbf{x}^j|^2. \tag{3.14}$$

In Eq. (3.14), the combination weights W_{ij} indicate the coefficients of the j-th point for the reconstruction of the i-th point. If $\mathbf{x}^j$ is not the neighbor of $\mathbf{x}^i$, then $W_{ij} = 0$. We also add the constraint $\sum_j W_{ij} = 1$. By solving the problem, one can finally get the optimal weights.
In the lower dimensional space, LLE reconstructs the neighbourhoods for all the sample points. During the reconstruction, the weights are preserved. To find the

lower dimensional representation, the following cost function is minimized:

$$E_1(\mathbf{Y}) = \sum_i |\mathbf{y}^i - \sum_i W_{ij}\mathbf{y}^j|^2. \tag{3.15}$$

The minimization problem is a quadratic expression of $\mathbf{y}^i$, which can be transformed to an eigenvalue problem. We assume

$$\mathbf{T} = (\mathbf{I} - \mathbf{W})^T(\mathbf{I} - \mathbf{W}), \tag{3.16}$$

where $\mathbf{I}$ is an identity matrix. Then the minimization problem can be written as

$$E_2(\mathbf{Y}) = \sum_{ij} \mathbf{T}_{ij}(\mathbf{y}^i \cdot \mathbf{y}^j). \tag{3.17}$$

We apply the following conditions:

$$\begin{aligned} &\sum_i \mathbf{y}^i = \mathbf{0}, \\ &\frac{1}{n}\sum_i \mathbf{y}^i(\mathbf{y}^i)^T = 1, \end{aligned} \tag{3.18}$$

and finally we solve the eigenvalue problem

$$\mathbf{TY} = \lambda\mathbf{Y}. \tag{3.19}$$

We choose the m eigenvectors which correspond to the m largest eigenvalues as the new coordinates in the lower-dimensional space.

3.5 Kernel principal component analysis

Compared with PCA, Kernel principal component analysis (KPCA) introduces a non-linear kernel function Ψ which maps the sample points to a higher dimensional space in order to handle non-linear data. Another key innovation is that KPCA applies a hypothesis: Any vector (even though a basis vector), can be represented by the linear combinations of all the samples in the space.
For a centred sample data $\mathbf{X} = [\mathbf{x}^1, \mathbf{x}^2, \cdots, \mathbf{x}^n]$, the kernel function Ψ is applied to project the sample data to a higher dimensional space in which the data can be linearly separated. We assume the dimensions of $\Psi(\mathbf{X})$ is H, then PCA is executed. Thus we have

$$\Psi(\mathbf{X})\Psi(\mathbf{X})^T\boldsymbol{\phi}_i = \lambda_i\boldsymbol{\phi}_i \tag{3.20}$$

where $\boldsymbol{\phi}_i, i = 1, 2, \cdots, M$ are the eigenvectors, and λ_i are the corresponding eigenvalues.
As mentioned above, the eigenvector $\boldsymbol{\phi}_i$ is expressed by the linear representations of sample data $\Psi(\mathbf{X})$, as follows:

$$\boldsymbol{\phi}_i = \sum_{j=1}^{n} \alpha_j \Psi(\mathbf{x}^j) = \Psi(\mathbf{X})\boldsymbol{\alpha}. \tag{3.21}$$

Substitute Eq. (3.21) to Eq. (3.20), we obtain

$$\Psi(\mathbf{X})\Psi(\mathbf{X})^T\Psi(\mathbf{X})\boldsymbol{\alpha} = \lambda_i\Psi(\mathbf{X})\boldsymbol{\alpha}. \tag{3.22}$$

we left-multiply both ends of Eq. (3.22) by $\Psi(\mathbf{X})^T$ and get

$$\Psi(\mathbf{X})^T\Psi(\mathbf{X})\Psi(\mathbf{X})^T\Psi(\mathbf{X})\boldsymbol{\alpha} = \lambda_i\Psi(\mathbf{X})^T\Psi(\mathbf{X})\boldsymbol{\alpha}. \tag{3.23}$$

In the next step, we construct $\boldsymbol{\Upsilon} = \Psi(\mathbf{X})^T\Psi(\mathbf{X})$. $\boldsymbol{\Upsilon}$ is assembled by the following form

$$\boldsymbol{\Upsilon} = \begin{bmatrix} \cdots & \cdots & \cdots \\ \cdots & \zeta(\mathbf{x}^i, \mathbf{x}^j) & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix}_{n\times n}. \tag{3.24}$$

$\zeta(\mathbf{x}^i, \mathbf{x}^j)$ is a kernel function. There are several popular kernel functions to choose, such as:
1) Linear kernel function:

$$\zeta(\mathbf{x}^i, \mathbf{x}^j) = \mathbf{x}^i \cdot \mathbf{x}^j; \tag{3.25}$$

2) Polynomial kernel function:

$$\zeta(\mathbf{x}^i, \mathbf{x}^j) = (\mathbf{x}^i \cdot \mathbf{x}^j + 1)^p; \tag{3.26}$$

3) Gaussian radial basis function:

$$\zeta(\mathbf{x}^i, \mathbf{x}^j) = \exp(-\frac{||\mathbf{x}^i - \mathbf{x}^j||}{\sigma^2}); \tag{3.27}$$

4) Sigmoid function:

$$\zeta(\mathbf{x}^i, \mathbf{x}^j) = \tanh(\mathbf{x}^i \cdot \mathbf{x}^j + \theta); \tag{3.28}$$

Consequently, Eq. (3.23) can be written as

$$\boldsymbol{\Upsilon}^2\boldsymbol{\alpha} = \lambda_i\boldsymbol{\Upsilon}\boldsymbol{\alpha}. \tag{3.29}$$

yielding

$$\boldsymbol{\Upsilon}\boldsymbol{\alpha} = \lambda_i\boldsymbol{\alpha}. \tag{3.30}$$

After solving this eigenvalue problem, we can easily select only several largest eigenvalues and their corresponding eigenvectors.
As any new data point $\mathbf{x}^{new}$ can be mapped to $\Psi(\mathbf{x}^{new})$ in the higher dimensional space, its new coordinates $\mathbf{y}^{new}$ after dimensionality reduction can be expressed as

$$\begin{aligned} \mathbf{y}^{new} &= \boldsymbol{\phi}_i^T\Psi(\mathbf{x}^{new}) \\ &= (\sum_{j=1}^{n}\alpha_j\Psi(\mathbf{x}^j))^T\Psi(\mathbf{x}^{new}) \\ &= (\Psi(\mathbf{X})\boldsymbol{\alpha})^T\Psi(\mathbf{x}^{new}) \\ &= \boldsymbol{\alpha}^T\Psi(\mathbf{X})^T\Psi(\mathbf{x}^{new}) \\ &= [\alpha_1, \alpha_2, \cdots, \alpha_n][\zeta(\mathbf{x}^1, \mathbf{x}^{new}), \zeta(\mathbf{x}^2, \mathbf{x}^{new}), \cdots, \zeta(\mathbf{x}^n, \mathbf{x}^{new})]^T. \end{aligned} \tag{3.31}$$

3.6 Testing examples

In this section, three examples including multiple cross-sections, the Swiss Roll and the S-Curve, are carried out with different manifold learning techniques proposed in previous sections of this chapter. We also discuss how the number of sample points will influence the learning results and the time costs with respect to the tested manifold learning algorithms.

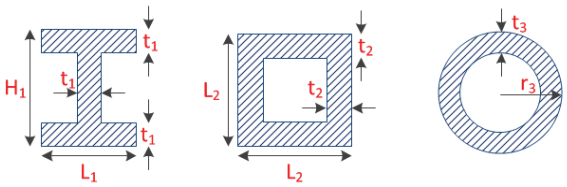


FIGURE 3.3: The three types of cross-sections

3.6.1 Multiple cross-section example

Here we propose to investigate the design instances of three kinds of cross-sections: the I-shape, the hollow square shape and the hollow circle shape as illustrated in Fig. 3.3. The geometrical parameters of all the shapes are listed as in Tab. 3.1, respectively. Due to the various of geometrical parameters, there are 99 cross-section instances: the I-shape numbered from 1 to 48, the hollow square numbered from 49 to 84 and the hollow circle numbered from 85 to 99.

TABLE 3.1: The available values of geometrical parameters

Parameter	Available values(m)
H_1	(0.05, 0.06, $\cdots$, 0.1)
L_1	(0.04, 0.05, $\cdots$, 0.09)
t_1	(0.05)
L_2	(0.04, 0.045, $\cdots$, 0.08)
t_2	(0.004, 0.005, $\cdots$, 0.007)
r_3	(0.04, 0.045, $\cdots$, 0.06)
t_3	(0.005, 0.006, 0.007)

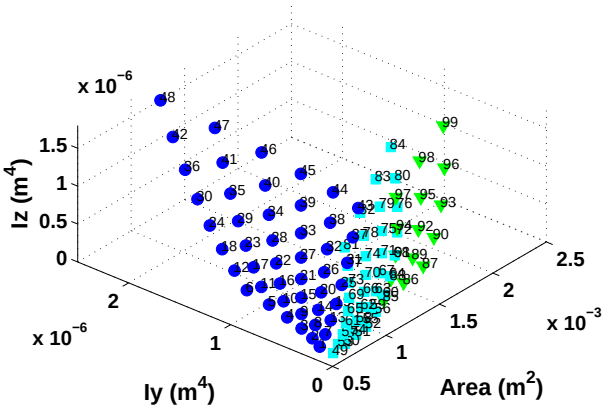


FIGURE 3.4: Cross-section instances

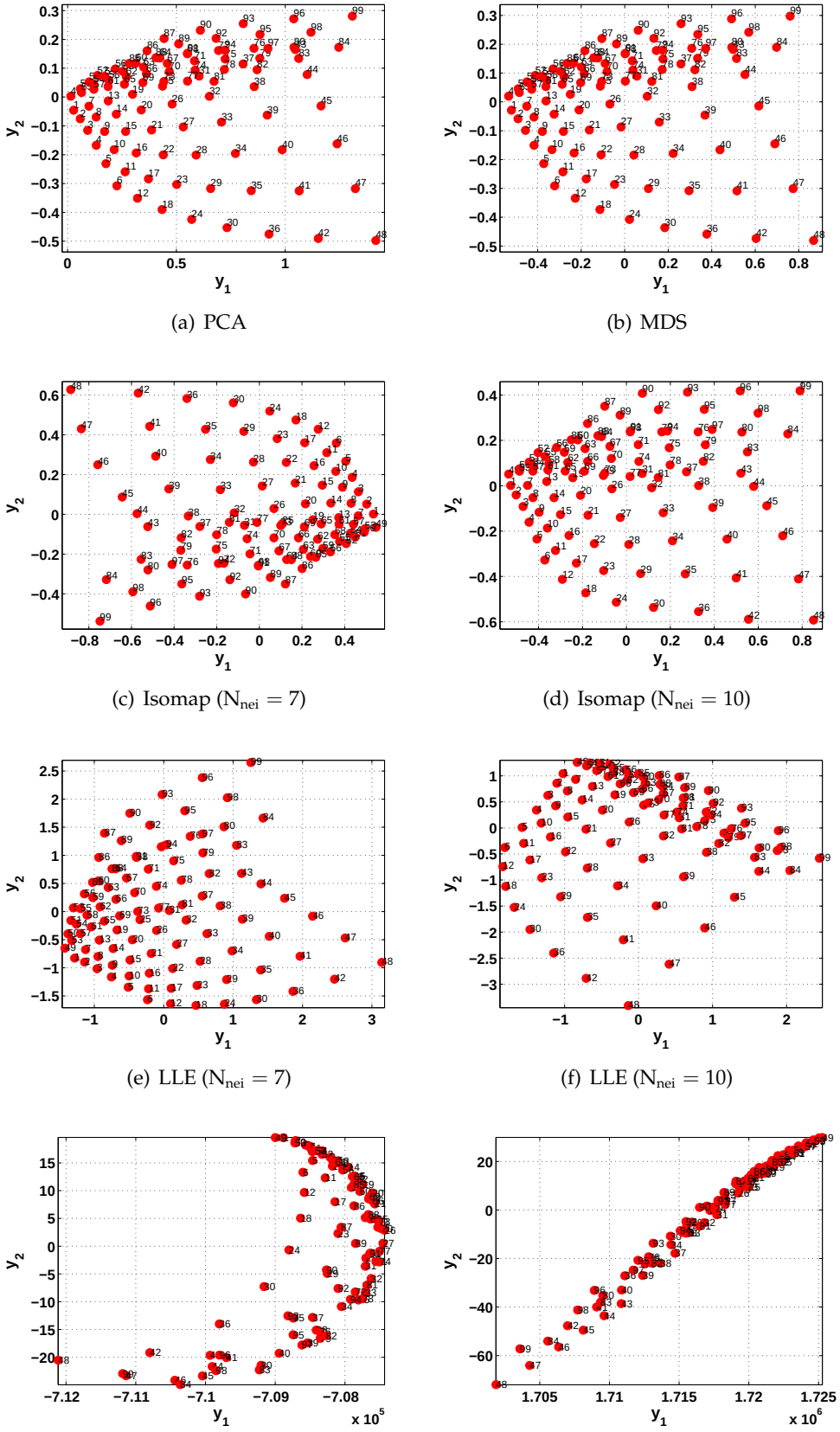
We take into consideration the physical properties including cross-section area, area moment of inertia along y-axis and area moment of inertia along z-axis (x-axis is recognized as the normal direction of the area). We display all the 99 bars in the 3-D physical space, as shown in Fig. 3.4, in which the round markers denote the instances of I-shape cross-sections, the square markers indicate the hollow square cross-sections and the triangle markers are the hollow circle cross-sections.

The investigated manifold learning methods include PCA, MDS, Isomap, LLE and KPCA. For both Isomap and LLE, we apply the k-means to determine the neighbour points, and the numbers of neighbour points are set as 7 and 10 respectively to illustrate how the choice will affect the learning results. Also, for KPCA, we apply two kinds of kernel functions: the Gaussian radial basis function ($\sigma = 1$) and polynomial kernel function ($p = 2$).

In Fig. 3.5(a-h), we display the learning results by PCA, MDS, Isomap, LLE and KPCA, respectively. While the original 99 design instances distributes in an approximate 2-D "Butterfly Wings" form, the Isomap and LLE, even the linear learning methods PCA and MDS can present a successful dimensionality reduction results. We can also find that for different learning parameters, the learning results obtained by Isomap and LLE show a small difference, but the main shape and the relative positions of the sample data are preserved. Unlike other manifold learning methods, KPCA with Gaussian radial basis function and polynomial function fail to preserve the general shape and the relative positions of the sample data of the 2-D space. The reason of this failure lies in the fact that although mapping non-linear data from initial space to higher dimensional space by using non-linear kernel functions makes data more easily divided with linear boundaries, the structure of samples' distribution may have been undermined. In the following sections, we will explain why KCPA with Gaussian RBF or polynomial functions may fail in data pattern preservation.

3.6.2 Swiss Roll example

The Swiss Roll example is a 2-D non-linear manifold embedded in 3-D space. As displayed in Fig. 3.6, 2000 sample points are gathered on the manifold to execute the dimensionality reduction. The investigated manifold learning methods including PCA, MDS, Isomap (7 and 10 neighbours), LLE (7 and 10 neighbours) and KPCA



(a) PCA (b) MDS

(c) Isomap ($N_{nei} = 7$) (d) Isomap ($N_{nei} = 10$)

(e) LLE ($N_{nei} = 7$) (f) LLE ($N_{nei} = 10$)

(g) KPCA (Gaussian RBF) (h) KPCA (Polynomials)

FIGURE 3.5: The learning results obtained by different methods and parameters (Cross-sections)

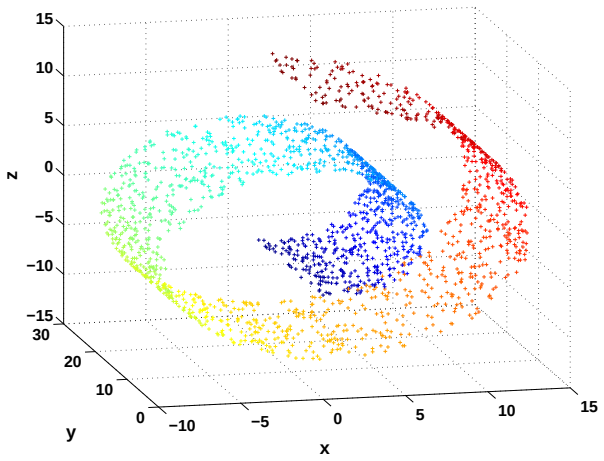
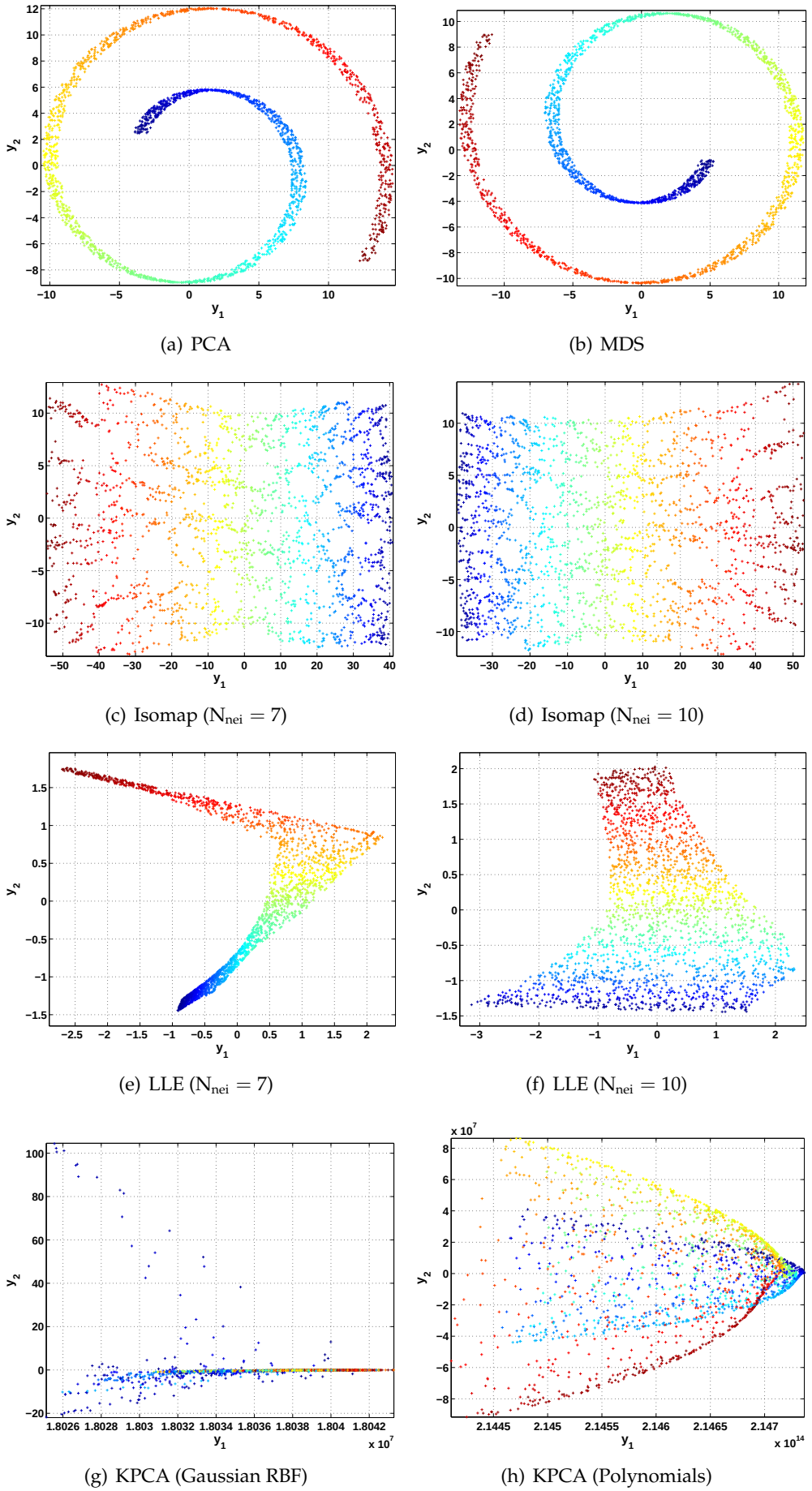


FIGURE 3.6: Sample points of the Swiss Roll

(Gaussian RBF and Polynomial kernel functions) are tested. As shown in Fig. 3.7, it is certain that the linear manifold learning methods: PCA and MDS can only adjust the projection directions thus the 2-D curled manifold cannot be unfolded in the reduced order space. Isomap with 7 or 10 neighbours points can generally unfold the the Swiss roll manifold. While LLE shows different results: The 10-neighbour group performs better in data pattern preservation, although the 7-neighbour group can also distinguish the two ends of the manifold; both groups bring a distortion of the regular plane manifold. When the number of neighbours increases, the distortion situation can be improved. KPCA performs as predicted: the Gaussian RBF ($\sigma = 1$) and Polynomial function (p $= 2$) cannot preserve the data configuration. And the reason will be illustrated in following section.

3.6.3 S-Curve example

We use the S-Curve example to test how the number of sample point influences the manifold learning results. As shown in Fig. 3.8, the S-Curve manifold is a curled plane embedded in 3-D space. We gather 1000, 2000, 3000 and 4000 sample points respectively to illustrate the effect. Fig. 3.9, 3.10, 3.11 and 3.12 show the manifold learning results of different methods with different numbers of samples points. The conclusion can be drawn that even 1000 sample points can support the manifold learning process being executed, but the success of unfolding the curled manifold relies on the mechanism of certain manifold learning methods. PCA and MDS still cannot unfold the manifold correctly, and their performance shows an honest fact that PCA and MDS can only deal with the cases that the potential manifold does not overlap along the projection direction (potentially) provided by PCA or MDS (Fig. 3.9-3.12(a,b)). The Isomap with 7 or 10 neighbours (Fig. 3.9-3.12(c,d)) can always obtain reasonable and successful results and behaves the best despite of different numbers of data points. LLE (Fig. 3.9-3.12(e,f))can also unfold the 2-D manifold in spite of that the corresponding results show some extent of distortion, and also the increasing of neighbour numbers can help ease the distortion. KPCA with Gaussian RBF ($\sigma = 1$) and Polynomial kernel functions (p $= 2$) still do not maintain the pattern of data points and present an incomplete unfolding of the 2-D manifold despite of the increasing number of sample points (Fig. 3.9-3.12(g,h)).



(a) PCA (b) MDS

(c) Isomap ($N_{nei} = 7$) (d) Isomap ($N_{nei} = 10$)

(e) LLE ($N_{nei} = 7$) (f) LLE ($N_{nei} = 10$)

(g) KPCA (Gaussian RBF) (h) KPCA (Polynomials)

FIGURE 3.7: The learning results obtained by different methods and parameters (Swiss Roll)

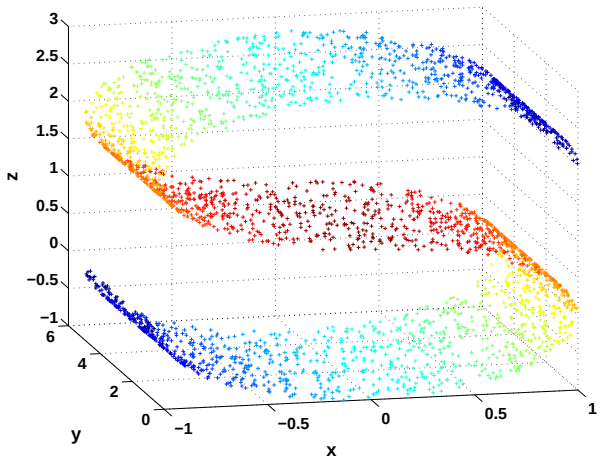


FIGURE 3.8: Sample points of the S-Curve

3.6.4 Time cost comparison

We also compare the efficiency of different manifold learning methods. We did a survey of time cost of different manifold learning methods with different numbers of sample points, namely 1000, 2000, 3000 and 4000. The tested example is also the S-Curve.
As listed in Tab. 3.2, we can find that as the numbers of sample points grow, the corresponding time consuming also rises for all methods. For the two linear learners, PCA costs less time than MDS although their learning results are totally the same. LLE performs efficiently compared to other non-linear learning methods, and the increase of neighbour numbers only brings a bit more time. Isomap is the most time consuming learning method, and its time cost is two orders of magnitude larger than that of LLE. Although KPCA seems to be faster than Isomap, it costs more time than LLE.
The advice for choosing manifold learning methods can be this: for linear cases, PCA is better than MDS considering the time cost; for non-linear cases, one can try LLE first, if LLE with large neighbour numbers show much distortion, then Isomap is the last choice to handle the manifold learning tasks.

TABLE 3.2: The corresponding learning time cost needed (Unit: s)

	1000 points	2000 points	3000 points	4000 points
PCA	7.5×10^{-4}	8.5×10^{-4}	9.2×10^{-4}	9.4×10^{-4}
MDS	5.2×10^{-1}	2.40×10^{0}	6.26×10^{0}	1.30×10^{1}
Isomap ($N_{nei} = 7$)	2.18×10^{1}	1.12×10^{2}	3.14×10^{2}	6.82×10^{2}
Isomap ($N_{nei} = 10$)	2.17×10^{1}	1.15×10^{2}	3.09×10^{2}	6.99×10^{2}

LLE ($N_{nei} = 7$)	1.3×10^{-1}	4.1×10^{-1}	7.4×10^{-1}	1.38×10^{0}
LLE ($N_{nei} = 10$)	1.5×10^{-1}	4.2×10^{-1}	8.1×10^{-1}	1.39×10^{0}
KPCA (Gaussian RBF)	1.37×10^{0}	6.85×10^{0}	1.58×10^{1}	3.20×10^{1}
KPCA (Polynomials)	4.1×10^{-1}	2.61×10^{0}	7.58×10^{0}	1.88×10^{1}

3.6.5 Discussion on KPCA

When Gaussian RBF ($\sigma = 1$) or polynomial ($p = 2$) is applied, the curled manifold hidden in higher dimensional space cannot be unfolded without much distortion. Now we would explain the reason.
We use Polynomial kernel function as an example. The Polynomial kernel function is written as:

$$\zeta(\mathbf{x}^i, \mathbf{x}^j) = (\mathbf{x}^i \cdot \mathbf{x}^j + 1)^p; \tag{3.32}$$

Then we withdraw the basic mode of the polynomials from Eq. 3.32 as

$$\zeta(\mathbf{x}^i, \mathbf{x}^j) = (\mathbf{x}^i \cdot \mathbf{x}^j)^p; \tag{3.33}$$

we can find Eq. 3.32 is just a linear combination of Eq. 3.33 with all-orders items not exceeding p.
We assume the dimensionality of the original observation space is 2, so we can write

$$\begin{aligned} \mathbf{x}^i &= [x^{i1} \quad x^{i2}]^T; \\ \mathbf{x}^j &= [x^{j1} \quad x^{j2}]^T; \end{aligned} \tag{3.34}$$

As we have defined that

$$\zeta(\mathbf{x}^i, \mathbf{x}^j) = \Psi(\mathbf{x}^i)^T \Psi(\mathbf{x}^j), \tag{3.35}$$

According to Eq. 3.33 and 3.35, we obtain

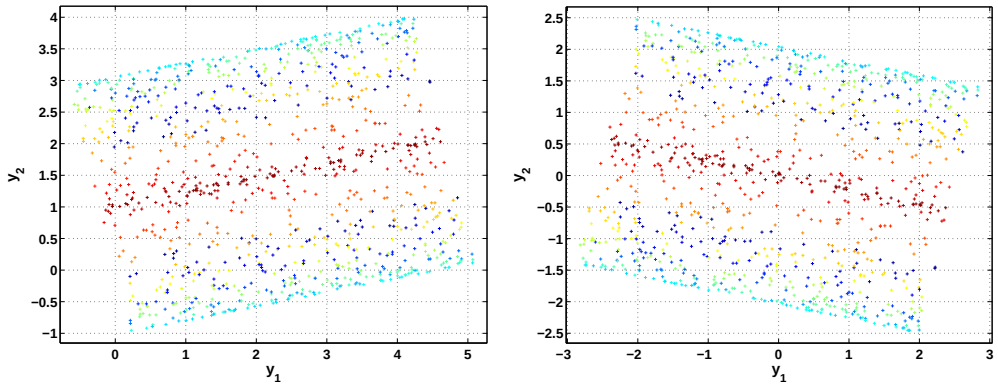
$$\begin{aligned} \Psi(\mathbf{x}^i) &= \Psi([x^{i1} \quad x^{i2}]^T) \\ &= [\sqrt{C_p^0}(x^{i1})^p \quad \sqrt{C_p^1}(x^{i1})^{p-1}x^{i2} \quad \sqrt{C_p^2}(x^{i1})^{p-2}(x^{i2})^2 \quad \cdots \quad \sqrt{C_p^p}(x^{i2})^p]^T \end{aligned} \tag{3.36}$$

where C_p^t is the combinatorial number, and we have $0 \leq t \leq p$.
As the dimension of $\Psi(\mathbf{x}^i)$ is p+1, this means we have mapped the original 2-D point $\mathbf{x}^i$ into a (p+1)-D space. Normally if the original dimension of sample points $\mathbf{x}^i$ is M, after the mapping of Ψ, the dimension of $\Psi(\mathbf{x}^i)$ will be C_{M+p-1}^p.
We would like to list the expanding results of the cases with p=1,2,3. When M equals 2 and p=1,

$$\Psi(\mathbf{x}^i) = [x^{i1} \quad x^{i2}]^T. \tag{3.37}$$

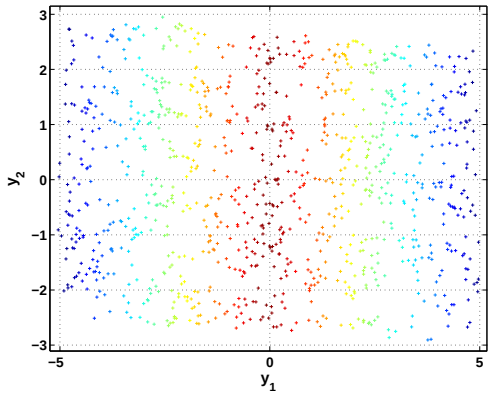
When M equals 2 and p=2,

$$\Psi(\mathbf{x}^i) = [(x^{i1})^2 \quad \sqrt{2}x^{i1}x^{i2} \quad (x^{i2})^2]^T. \tag{3.38}$$

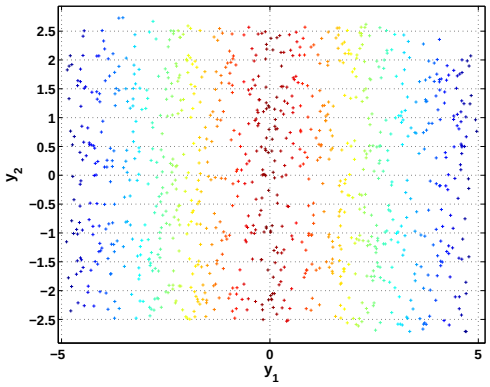


(a) PCA

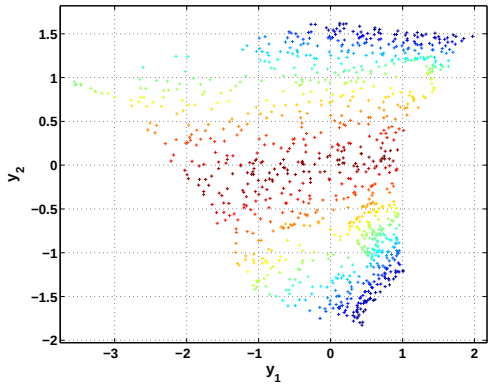
(b) MDS



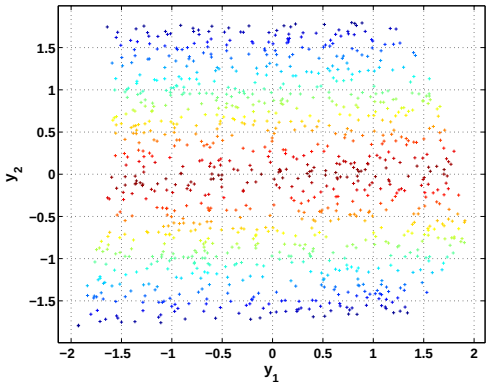
(c) Isomap ($N_{nei} = 7$)



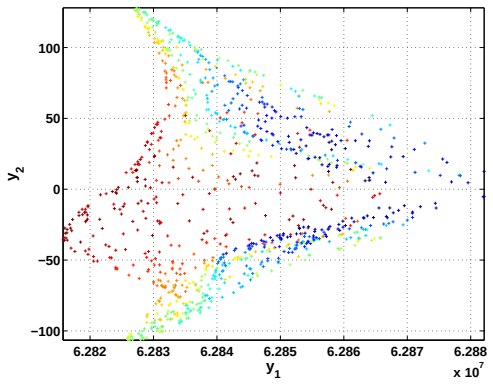
(d) Isomap ($N_{nei} = 10$)



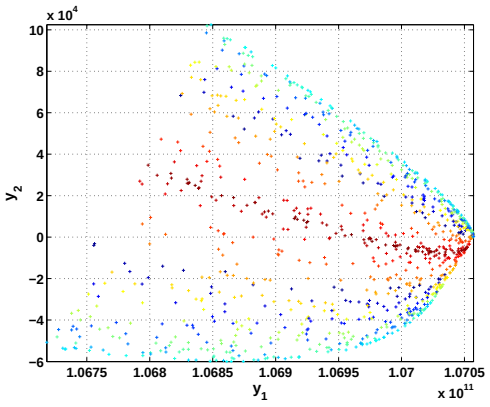
(e) LLE ($N_{nei} = 7$)



(f) LLE ($N_{nei} = 10$)

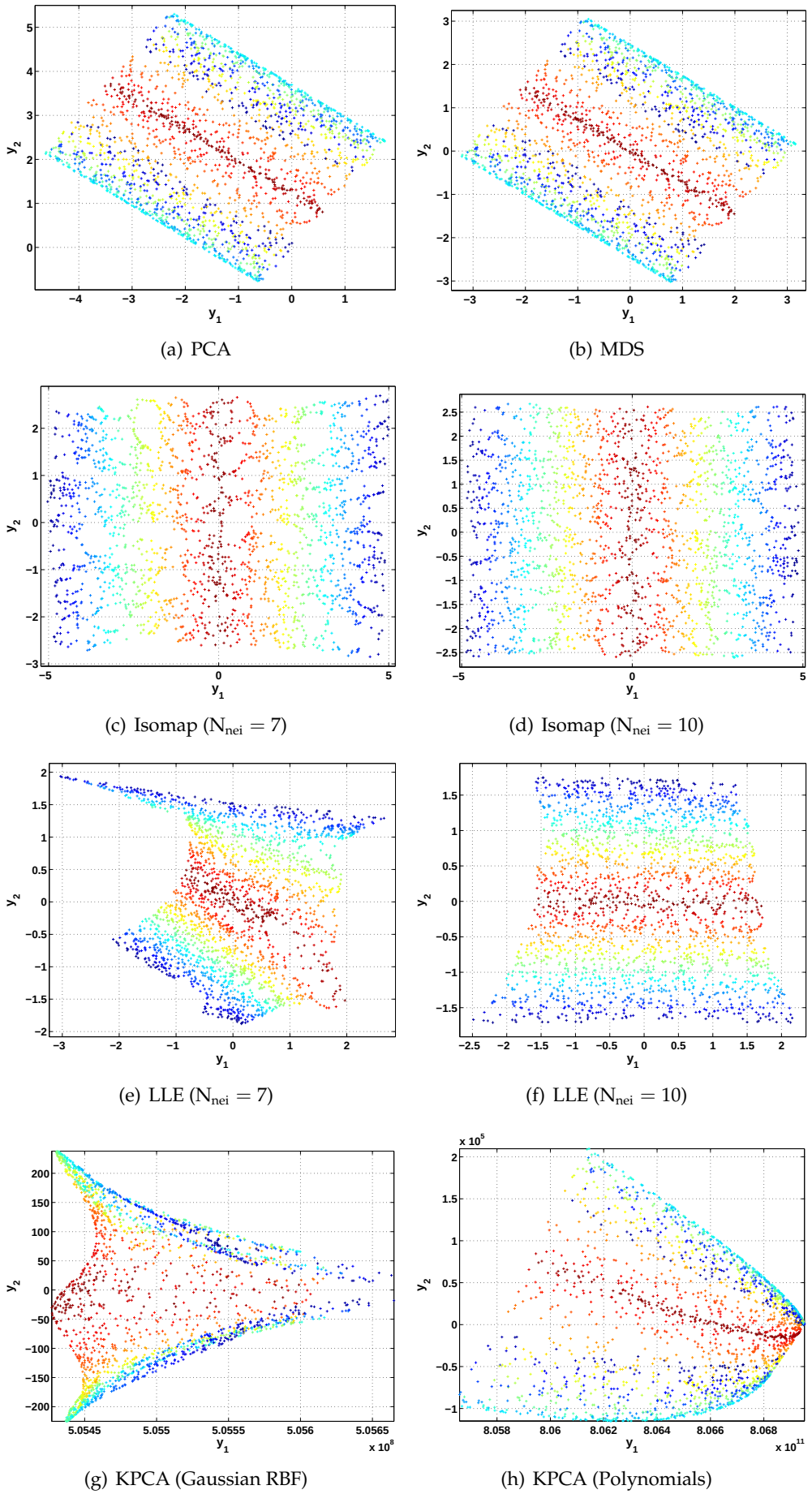


(g) KPCA (Gaussian RBF)



(h) KPCA (Polynomials)

FIGURE 3.9: The learning results obtained by different methods and parameters (S-Curve) with 1000 sample points



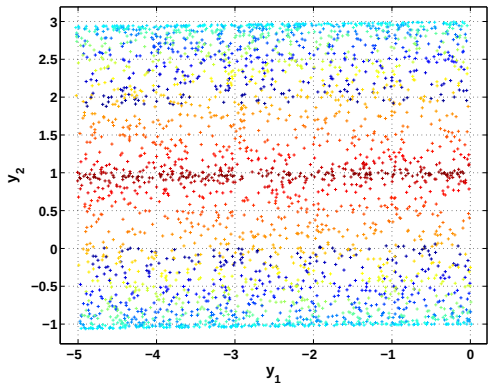
(a) PCA (b) MDS

(c) Isomap ($N_{nei} = 7$) (d) Isomap ($N_{nei} = 10$)

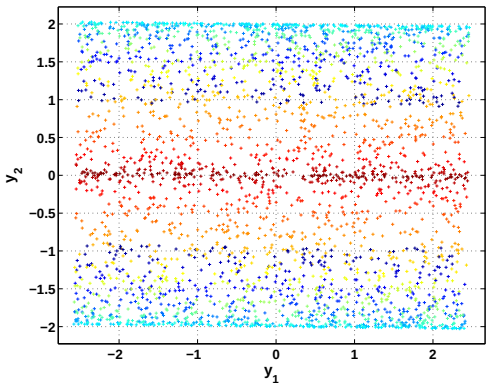
(e) LLE ($N_{nei} = 7$) (f) LLE ($N_{nei} = 10$)

(g) KPCA (Gaussian RBF) (h) KPCA (Polynomials)

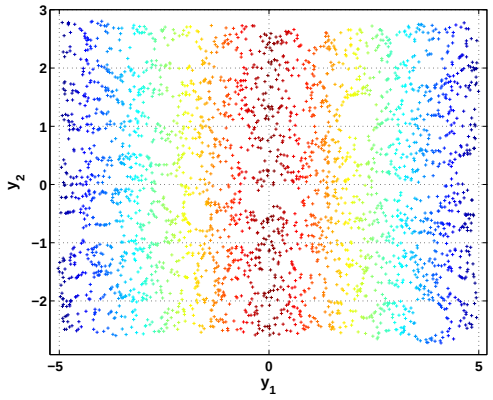
FIGURE 3.10: The learning results obtained by different methods and parameters (S-Curve) with 2000 sample points



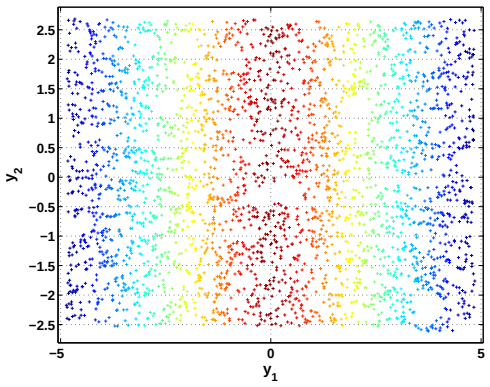
(a) PCA



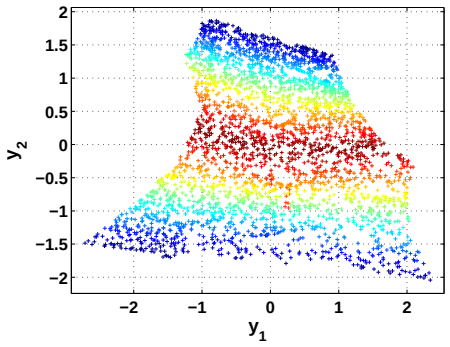
(b) MDS



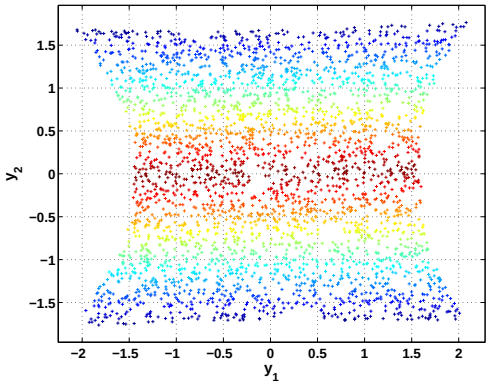
(c) Isomap ($N_{nei} = 7$)



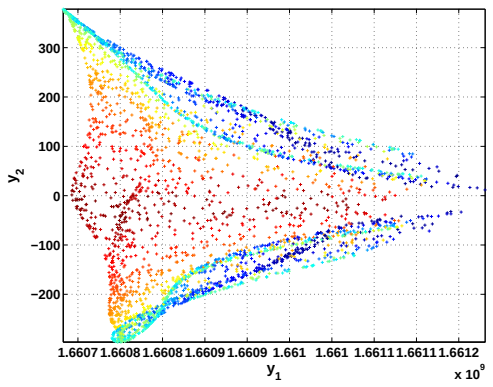
(d) Isomap ($N_{nei} = 10$)



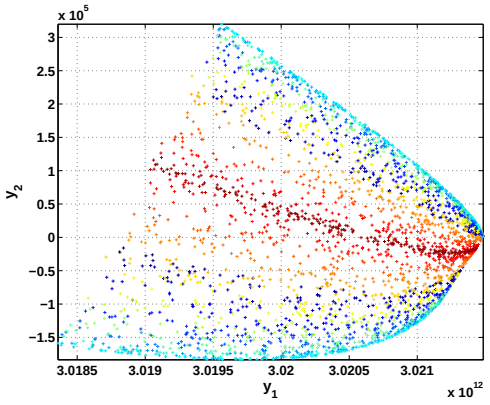
(e) LLE ($N_{nei} = 7$)



(f) LLE ($N_{nei} = 10$)

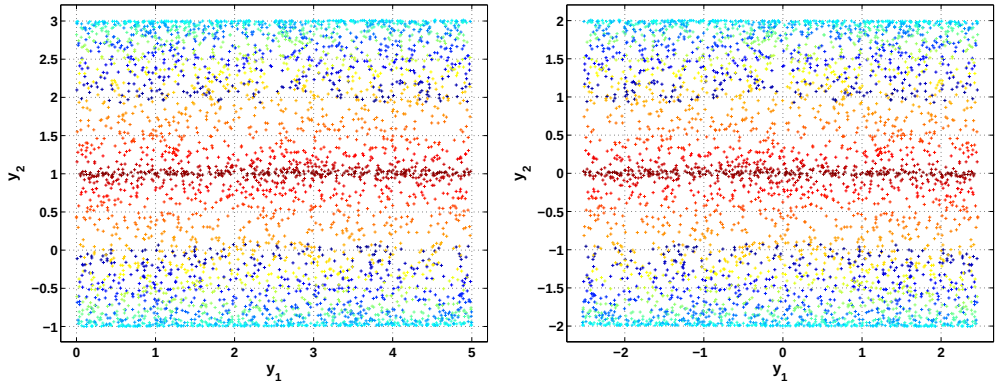


(g) KPCA (Gaussian RBF)



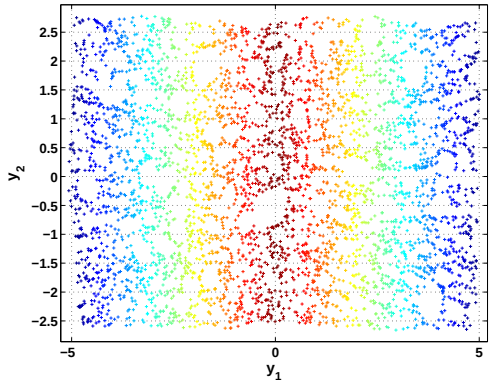
(h) KPCA (Polynomials)

FIGURE 3.11: The learning results obtained by different methods and parameters (S-Curve) with 3000 sample points

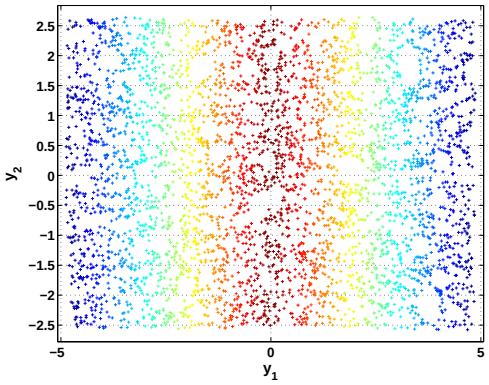


(a) PCA

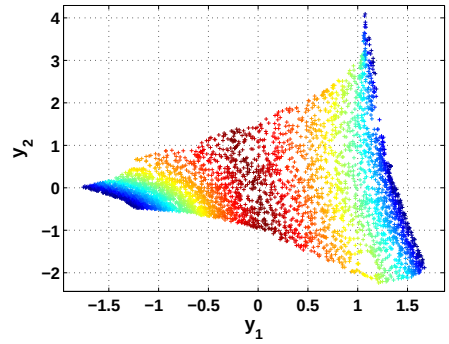
(b) MDS



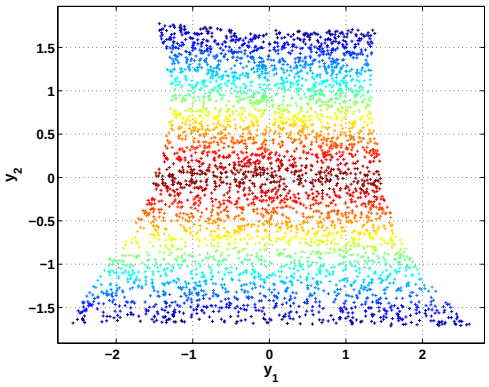
(c) Isomap ($N_{nei} = 7$)



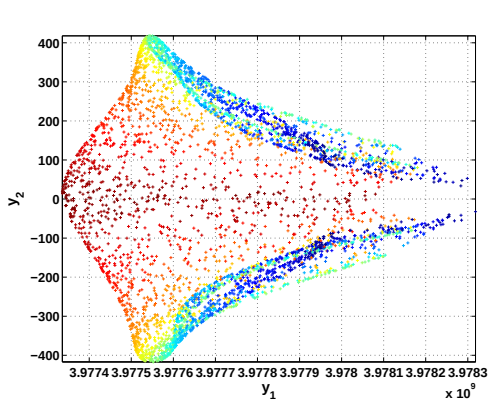
(d) Isomap ($N_{nei} = 10$)



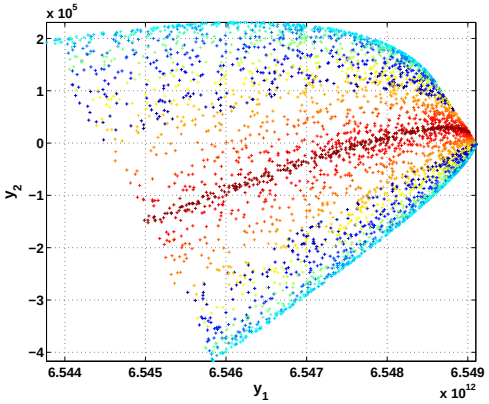
(e) LLE ($N_{nei} = 7$)



(f) LLE ($N_{nei} = 10$)



(g) KPCA (Gaussian RBF)



(h) KPCA (Polynomials)

FIGURE 3.12: The learning results obtained by different methods and parameters (S-Curve) with 4000 sample points

When M equals 2 and p=3,

$$\Psi(\mathbf{x}^i) = [(x^{i1})^3 \quad \sqrt{3}(x^{i1})^2 x^{i2} \quad \sqrt{3}x^{i1}(x^{i2})^2 \quad (x^{i2})^3]^T. \tag{3.39}$$

We can see that if p=1, KPCA degenerates to be PCA; while if $p \geq 2$, KPCA becomes non-linear.
Next we will discuss KPCA with those non-linear mappings from the respects of data distance, angle changes.

Data distance

The distances between sample points are critically important in many manifold learning methods, for example MDS and Isomap. Even in LLE, distances between sample points are used to determine the local neighbourhood, helping construct the local weights.
We first take the case of p=2 and M=2as an example. We assume there are two sample points $\mathbf{x}^1$ and $\mathbf{x}^2$, and they can be expressed as

$$\begin{aligned} \mathbf{x}^1 &= [x^{11} \quad x^{12}]^T; \\ \mathbf{x}^2 &= [x^{21} \quad x^{22}]^T. \end{aligned} \tag{3.40}$$

We define the vector $\mathbf{L}_0$ as

$$\begin{aligned} \mathbf{L}_0 &= \mathbf{x}^2 - \mathbf{x}^1 \\ &= [x^{21} - x^{11} \quad x^{22} - x^{12}]^T. \end{aligned} \tag{3.41}$$

As already written in Eq. 3.38, we can also define the vector $\mathbf{L}_1$ as

$$\begin{aligned} \mathbf{L}_1 &= \Psi(\mathbf{x}^2) - \Psi(\mathbf{x}^1) \\ &= [(x^{21})^2 - (x^{11})^2 \quad (x^{22})^2 - (x^{12})^2 \quad \sqrt{2}(x^{21}x^{22} - x^{11}x^{12})]^T. \end{aligned} \tag{3.42}$$

In order to handle the inner product, we reformulate $\mathbf{L}_0$ in 3-D space

$$\mathbf{L}_0 = [x^{21} - x^{11} \quad x^{22} - x^{12} \quad 0]^T. \tag{3.43}$$

If the distance between the two sample points $\mathbf{x}^1$ and $\mathbf{x}^2$ can be preserved after mapping, namely $||\mathbf{L}_0|| = ||\mathbf{L}_1||$, then the following equation must hold

$$\Delta^2 = (\mathbf{L}_1 + \mathbf{L}_0) \cdot (\mathbf{L}_1 - \mathbf{L}_0) = 0. \tag{3.44}$$

where Δ^2 indicates discriminant when the investigated polynomial order is 2.
Substitute Eq. 3.42 and 3.43 into Eq. 3.44, we finally obtain

$$\Delta^2 = \Delta_1^2 + \Delta_2^2 \tag{3.45}$$

in which

$$\begin{aligned} \Delta_1^2 &= (x^{21} - x^{11})^2((x^{21} + x^{11})^2 - 1) + (x^{22} - x^{12})^2((x^{22} + x^{12})^2 - 1), \\ \Delta_2^2 &= 2(x^{21}x^{22} - x^{11}x^{12})^2. \end{aligned} \tag{3.46}$$

If Δ^2 equals 0, it means the distance between the two chosen points remains unchanged, which means the distance between data points can be preserved during

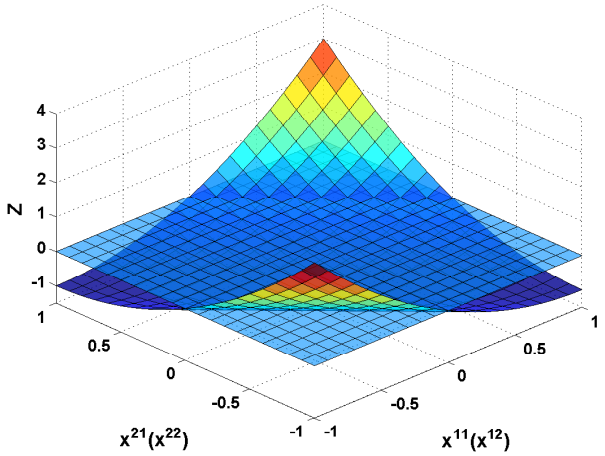


FIGURE 3.13: The values of z versus x^{21} and x^{11} or x^{22} and x^{12}

the mapping; if Δ^2 is positive, the length of $\mathbf{L}_1$ is larger than that of $\mathbf{L}_0$, which indicates the point-wise distance is lengthened; if Δ^2 is negative, it indicates the distance between the two points becomes shorter, it means the mapping actually shortens the original distance.
We can also find if $z = (x^{21} + x^{11})^2 - 1 > 0$ and $z = (x^{22} + x^{12})^2 - 1 > 0$, Δ_1^2 will then be positive, consequently Δ^2 will be also positive. The same result occurs to Here we plot z in the domain of [-1 1], as shown in Fig. 3.13. In Eq. 3.46, Δ_1^2 is named the sign-uncertain item because Δ_1^2 is possible to be positive, 0 or negative; while Δ_2^2 is a sign-certain item because it always remains non-negative.
When p comes to 3, we can also write

$$\begin{aligned}
\Delta^3 &= \Delta_1^3 + \Delta_2, \\
\Delta_1^3 &= (x^{21} - x^{11})^2(((x^{21})^2 + (x^{11})^2 + x^{21}x^{11})^2 - 1) \\
&\quad + (x^{22} - x^{12})^2(((x^{22})^2 + (x^{12})^2 + x^{22}x^{12})^2 - 1), \\
\Delta_2^3 &= 3((x^{21})^2x^{22} - (x^{11})^2x^{12})^2 + 3((x^{22})^2x^{21} - (x^{12})^2x^{11})^2.
\end{aligned} \tag{3.47}$$

We can also find that Δ_2^3 is always non-negative, but Δ_1^3 still has the uncertainty on it sign. The values of z are displayed in the Fig. 3.14. We give a generally form of the discriminant Δ^p with polynomial order p and original data dimension M=2 on how the polynomial kernel function affects the data distances:

$$\begin{aligned}
\Delta^p &= \Delta_1^p + \Delta_2^p, \\
\Delta_1^p &= \sum_{i=1}^{2}(x^{2i} - x^{1i})^2((\frac{(x^{2i})^p - (x^{1i})^p}{x^{2i} - x^{1i}})^2 - 1), \\
\Delta_2^p &= \sum_{j=1}^{p-1} C_p^j((x^{21})^j(x^{22})^{p-j} - (x^{11})^j(x^{12})^{p-j}).
\end{aligned} \tag{3.48}$$

We can see that the sigh of the discriminant is not only a function of the polynomial order p, it is also heavily dependent on the coordinates of the two points. Only when p=1, the relation $\Delta^1 = 0$ always holds despite of point coordinates. For the case p>1, the positions of two points play a key role to determine the sign of Δ^p
When we apply polynomial kernel functions with multiple orders, for example all-order items lower than or equal to p, then the overall discriminant becomes a linear

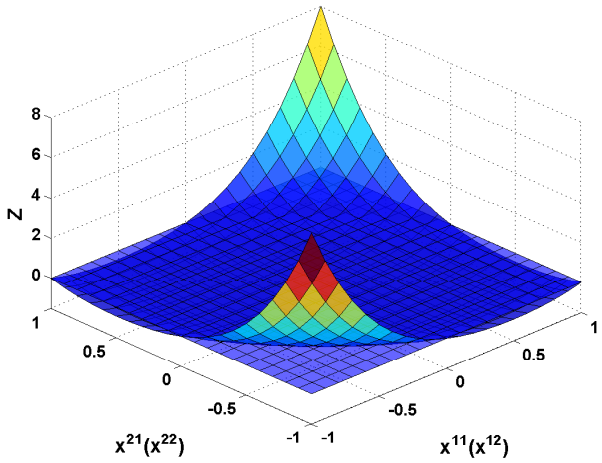


FIGURE 3.14: The values of z versus x^{21} and x^{11} or x^{22} and x^{12}

combination of the component discriminants with every order. And the combination coefficients remain the same as those of the polynomial coefficients.
If we consider the function Δ^p as a field function of order number p, then the overall discriminant will be the superposition of overall field functions Δ^p.
It is important to not that a p order polynomial can only maps the original data into a limited dimensional space. The Gaussian RBF can map the data into an infinite dimensional space because the Gaussian RBF is able to be expended into the form of polynomials with infinite items. And the parameter of Gaussian RBF σ control the coefficients of different polynomial items: when σ is large, the coefficients of high-dimensional polynomials will attenuate quickly, as a result, the high-dimensional polynomial items become secondary or negligible; when σ is a small number, the high-dimensional polynomial items seem to be considerable.

Angle change

Angle preservation is also an important issue in manifold learning field since several manifold learning techniques are developed based on angle preservation from higher dimensional space to lower dimensional space. Here we will demonstrate how the angle changes during the lower-to higher dimensional space mapping. Suppose the original dimension number is 2 and three data points are given as following:

$$\begin{aligned} \mathbf{x}^1 &= \begin{bmatrix} x^{11} & x^{12} \end{bmatrix}^T; \\ \mathbf{x}^2 &= \begin{bmatrix} x^{21} & x^{22} \end{bmatrix}^T. \\ \mathbf{x}^3 &= \begin{bmatrix} x^{31} & x^{32} \end{bmatrix}^T. \end{aligned} \tag{3.49}$$

We define two vectors:

$$\begin{aligned} \mathbf{L}_0^{12} &= \mathbf{x}^2 - \mathbf{x}^1 = \begin{bmatrix} x^{21} - x^{11} & x^{22} - x^{12} \end{bmatrix}^T, \\ \mathbf{L}_0^{13} &= \mathbf{x}^3 - \mathbf{x}^1 = \begin{bmatrix} x^{31} - x^{11} & x^{32} - x^{12} \end{bmatrix}^T \end{aligned} \tag{3.50}$$

Then the angle θ_0 between $\mathbf{L}_0^{12}$ and $\mathbf{L}_0^{13}$ satisfies:

$$\begin{aligned} cos(\theta_0) &= \frac{\mathbf{L}_0^{12} \cdot \mathbf{L}_0^{13}}{||\mathbf{L}_0^{12}||||\mathbf{L}_0^{13}||} \\ &= \frac{(x^{21} - x^{11})(x^{31} - x^{11}) + (x^{22} - x^{12})(x^{32} - x^{12})}{\sqrt{(x^{21} - x^{11})^2 + (x^{22} - x^{12})^2}\sqrt{(x^{31} - x^{11})^2 + (x^{32} - x^{12})^2}} \end{aligned} \tag{3.51}$$

We take p = 2 polynomial function as an example, then it is easily obtained

$$\begin{aligned} \mathbf{L}_1^{12} &= \Psi(\mathbf{x}^2) - \Psi(\mathbf{x}^1) \\ &= [(x^{21})^2 - (x^{11})^2 \quad (x^{22})^2 - (x^{12})^2 \quad \sqrt{2}(x^{21}x^{22} - x^{11}x^{12})]^T; \\ \mathbf{L}_1^{13} &= \Psi(\mathbf{x}^3) - \Psi(\mathbf{x}^1) \\ &= [(x^{31})^2 - (x^{11})^2 \quad (x^{32})^2 - (x^{12})^2 \quad \sqrt{2}(x^{31}x^{32} - x^{11}x^{12})]^T; \end{aligned} \tag{3.52}$$

So we can write the new angle θ_1 between $\mathbf{L}_1^{12}$ and $\mathbf{L}_1^{13}$ satisfies:

$$\begin{aligned} cos(\theta_1) &= \frac{\mathbf{L}_1^{12} \cdot \mathbf{L}_1^{13}}{||\mathbf{L}_1^{12}||||\mathbf{L}_1^{13}||} \\ &= \frac{((x^{21})^2 - (x^{11})^2)((x^{31})^2 - (x^{11})^2) + ((x^{22})^2 - (x^{12})^2)((x^{32})^2 - (x^{12})^2) + }{\sqrt{((x^{21})^2 - (x^{11})^2)^2 + ((x^{22})^2 - (x^{12})^2)^2 + (\sqrt{2}(x^{21}x^{22} - x^{11}x^{12}))^2} \times} \\ &\quad \frac{(\sqrt{2}(x^{21}x^{22} - x^{11}x^{12}))(\sqrt{2}(x^{31}x^{32} - x^{11}x^{12}))}{\sqrt{((x^{31})^2 - (x^{11})^2)^2 + ((x^{32})^2 - (x^{12})^2)^2 + (\sqrt{2}(x^{31}x^{32} - x^{11}x^{12}))^2}} \end{aligned} \tag{3.53}$$

From Eq. 3.51 and 3.53, we can find that the angle cannot be preserved during the mapping in most cases except for several special coincidences. One of the reason is that the angle θ_1 depends heavily on the coordinates of sample points.
In order reveal how the angle changes, we introduce an extra conditions to simplify the expression of the two angles. The introduced condition is: the point $\mathbf{x}^1$ locates at the origin, namely $x^{11} = 0$ and $x^{12} = 0$. Then Eq. 3.51 and 3.53 can be simplified as following

$$\begin{aligned} cos(\theta_0) &= \frac{x^{21}x^{31} + x^{22}x^{32}}{\sqrt{(x^{21})^2 + (x^{22})^2}\sqrt{(x^{31})^2 + (x^{32})^2}}, \\ cos(\theta_1) &= \frac{(x^{21}x^{31} + x^{22}x^{32})^2}{((x^{21})^2 + (x^{22})^2)((x^{31})^2 + (x^{32})^2)}. \end{aligned} \tag{3.54}$$

Thus we yield

$$cos(\theta_1) = (cos(\theta_0))^2. \tag{3.55}$$

Fig.3.15 shows the relationship between θ_0 and θ_1. In this figure, we can find that when θ_0 is close to 0 and 0.5π, θ_1 is approximately the same as θ_0. When $0 \leq \theta_0 \leq 0.5\pi$, there is some differences between θ_0 and θ_1. When $0.5\pi \leq \theta_0 \leq \pi$, the obtuse angle θ_0 has been transformed to an acute angle θ_1. Obviously, this is a folding operation instead of an unfolding operation and it is harmful to unfold a potential manifold. So we can say the angle in the original space cannot be preserved in the higher dimensional space unless the original angle is 0 or 0.5π.
Note that when the point $\mathbf{x}^1$ is not the origin, Eq. 3.55 does not hold as a result. Thus even when θ_0 equals 0 (which indicates the three points are all along one line in the 2-D space), the corresponding θ_1 will be probably non-zero. It means that a 2-D line in the 2-D space will be mapped to a curve in the 3-D space: an linear object now

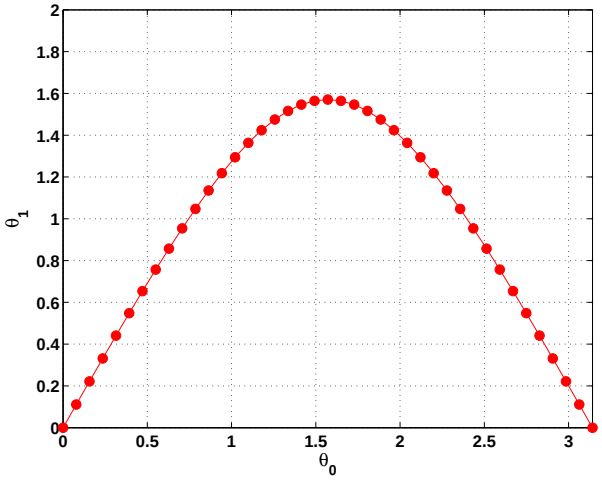


FIGURE 3.15: The relationship between θ_0 and θ_1

becomes non-linear. In other words, the intrinsic dimensionality of the manifold increases after the mapping.

Conclusions on KPCA

As we have obtained, if the polynomial order p=1, then KPCA degenerates to PCA, the point distances, point angles and linear manifold can be thus preserved. When the polynomial order p>1, the preservation of the point distances and angles is lost. The lower dimensional, linear, unfolded, uncurled manifold will become a higher dimensional, non-linear, folded, curled manifold. This can be the explanation of why KPCA using Gaussian RBF or Polynomials as the kernel function is probably fail to reduce the dimensionality.
But it is not suitable to draw the conclusion that KPCA is not able to handle the dimensionality reduction problem, because KPCA is a systematic manifold learning method which contains various kinds of kernel function. There are already some researches which have proved that MDS, Isomap and LLE are also KPCA with unique kernel functions, and sometimes they work rather well for non-linear manifold learning problems. The existence of new kernel functions which allows KPCA succeeding in non-linear manifold learning remains to be a question, and is worthy for researchers to explore.

Chapter 4

Categorical optimization: a discrete evolutionary approach

4.1 Background

In structural optimization problems, including size, shape, topology optimization (Kaveh and Talatahari, 2009; Sokolowski and Zolesio, 1992; Bendsøe and Kikuchi, 1988; Bendsoe and Sigmund, 2013) or material selection problems (Ashby and Johnson, 2013; Gao and Zhang, 2011), the design variables can be generally classified into two main groups: continuous and non-continuous ones. The former can take any value within a real interval $[x_{min}, x_{max}]$, and can be tackled with gradient-based algorithms provided that all functions are continuous and differentiable.
A further classification of non-continuous variables is advocated in two main types: discrete (including the integer) and categorical ones (Coelho et al., 2015; Sloane and Morgan, 1996; Lee and Kim, 2010b; McCane and Albert, 2008) which take non-numerical values. If the categorical variables cannot be ordered, they are named nominal variables, for instance, the choice of a material or the selection of a beam cross-section type (square, I-shaped, round,...) within a catalog.
The representation of such variables is a central issue (Coelho et al., 2015; Filomeno Coelho, 2014). To address this matter, several coding schemes have been proposed in the literature, including binary, real and simplex. Binary coding represents the values of non-continuous variables as strings of $0/1$ digits (Goldberg, 2006). Real coding applies the "one gene, one variable" principle and represents the values of variables through real numbers (Herrera, Lozano, and Verdegay, 1998). Simplex coding is an alternative to binary or real coding. First applied in pattern classification (Fu, Yan, and Huang, 2008), simplex has been developed to encode unordered values of categorical variables (Filomeno Coelho, 2014; Herrera et al., 2014; Coelho et al., 2015; Filomeno Coelho, 2012). Instead of using binary strings or real numbers, simplex exploits the coordinate vectors of points to represent the categorical design candidates in search space, in which the Euclidean distances between each pair of points are kept the same. There are also several optimization methods which are tailored to deal with categorical variables, including the mesh adaptive direct search (Abramson et al., 2009), ant colony (Liao et al., 2014), mixed variable programming (Kokkolaras, Audet, and Dennis, 2001) and global descent approach (Lindroth and Patriksson, 2011).
However, when the coding methods introduced are applied to non-ordinal categorical variables, some delicate issues arise. The first problem is that the binary and real encoding techniques are not unique, which will affect both the optimization process and results. Another concern is the loss of physical meaning. Simplex coding can avoid the issue of non-uniqueness, however, its representation dimensionality is high when the design possibilities are large.

In this chapter, we propose a three-step approach to handle the categorical optimization problem. First, we use sets of multi-dimensional attributes to represent categorical variables. Then, since these attributes may be correlated with each other or even mutually dependent, it is necessary to eliminate the possible redundant dimensionality. So we introduce an intrinsic dimensionality estimation criterion and manifold learning technique to represent the high-dimensional data in terms of a low-dimensional graph. Finally, we propose custom-built evolutionary operators acting on the graph obtained in the previous step to execute the optimization process.

The two well-known manifold learning techniques are the Principal Component Analysis (PCA) (Jolliffe, 2002) and Multi-Dimensional Scaling (MDS) (Martin and Eroglu, 1993). PCA takes advantage of eigenvalue analysis to discover a lower dimensional representation which preserves the maximum variance information. MDS concentrates on preserving the Euclidean distances during the mapping from higher dimensional space to lower dimensional space. These are two equivalent methods, and work well in linear cases, but may give unsatisfactory results when handling non-linear data. Isomap (Tenenbaum, De Silva, and Langford, 2000) replaces the Euclidean distance by the geodesic distance and then runs the classical MDS to find the centralized lower dimensional representation, enabling to deal with the nonlinear case. These manifold learning methods have been already applied in mechanics (Meng et al., 2016), including identification (Meng et al., 2015) and structural optimization (Raghavan et al., 2013).

Based on the reduced-order representation obtained by Isomap, the evolutionary operators, including crossover and mutation operators, are developed in the present work by using the shortest paths and nearest neighbours, respectively. Then, these operators are applied to genetic algorithms to execute the optimization tasks.

We set up the chapter as follows. In section 4.2 we explain the multi-dimensional discrete representation for the categorical variables. In section 4.3, we illustrate how Isomap works to reduce the dimensionality and introduce the intrinsic dimensionality estimation criterion. In section 4.4, the graph-based evolutionary operators, including crossover and mutation, are developed. In section 4.5, the proposed approach is tested with three examples. Finally, in section 4.6, the conclusions and prospects are made.

4.2 Representation of categorical variables

A categorical variable can take values within a set of predefined non-numerical instances. In statistics, categorical variables are usually applied to mark different individuals with a list of categories, while in computer science or mathematics, they correspond to enumerations. In general, the possible values for categorical variables are referred as instances or levels (Banker and Morey, 1986) in optimization research field. In the present study, we focus on non-ordinal categorical variables (Fig. 4.1). In an engineering application, we propose to consider the instances of categorical variables as discrete points in the multi-dimensional space of physical features. For example, colors can be treated as RGB triples, the materials can be represented as lists of constitutive properties, etc.

Consider a categorical variable which is represented by vector $\mathbf{x}$ and takes several possible instances $\mathbf{x}^j, j = 1, 2, \cdots, n$. An instance can be represented by a set of attributes:

$$\mathbf{x}^j = ({}^1a^j, {}^2a^j, \cdots, {}^Ma^j)^T \tag{4.1}$$

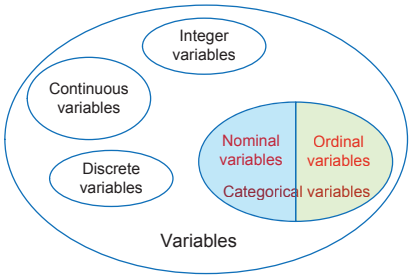


FIGURE 4.1: Classification of design variables

TABLE 4.1: A catalog of bars' physical features

Label	^{1}a: Area (m^2)	^{2}a: Young's modulus (Pa)	^{3}a: Density (kg/m^3)
$\mathbf{x}^1$	$^1a^1$=7.28e-4	$^2a^1$=2.07e11	$^3a^1$=7830
$\mathbf{x}^2$	$^1a^2$=8.00e-4	$^2a^2$=1.64e11	$^3a^2$=8270
$\mathbf{x}^3$	$^1a^3$=8.66e-4	$^2a^3$=2.20e11	$^3a^3$=7930
$\mathbf{x}^4$	$^1a^4$=8.66e-4	$^2a^4$=1.85e11	$^3a^4$=8440
$\mathbf{x}^5$	$^1a^5$=9.00e-4	$^2a^5$=1.78e11	$^3a^5$=8890
$\mathbf{x}^6$	$^1a^6$=9.39e-4	$^2a^6$=2.07e11	$^3a^6$=7830
$\mathbf{x}^7$	$^1a^7$=1.00e-3	$^2a^7$=1.64e11	$^3a^7$=8270
$\mathbf{x}^8$	$^1a^8$=1.28e-3	$^2a^8$=2.07e11	$^3a^8$=7830
$\mathbf{x}^9$	$^1a^9$=1.32e-3	$^2a^9$=1.85e11	$^3a^9$=8440

where $^k a^j$ stands for the k-th attribute of the j-th instance, and M indicates the dimensionality of categorical variable $\mathbf{x}$. Noting that the attributes can be also regarded as the coordinates of an instance, we can represent all the instances by a set of discrete points in R^M.
Consider a truss structure consisting of bars. Every bar can be represented by three physical properties: cross-section area, Young's modulus and density. Tab. 4.1 gives 9 possible instances for a bar, and a graphical interpretation of the 9 instances is given in Fig. 4.2. We will use this example in the following section to illustrate the dimensionality reduction.

4.3 Redundant dimensionality reduction

In general application, the dimensionality of attributes may be high, which causes low optimization efficiency. We find that sometimes these attributes are correlated with each other, or mutually dependent. For example, for a linear elastic material problem, the Young's modulus E, the Poisson's ratio ν and the shear modulus G are

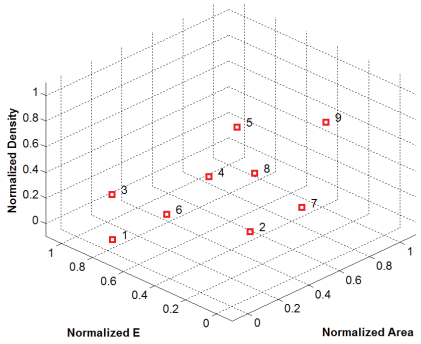
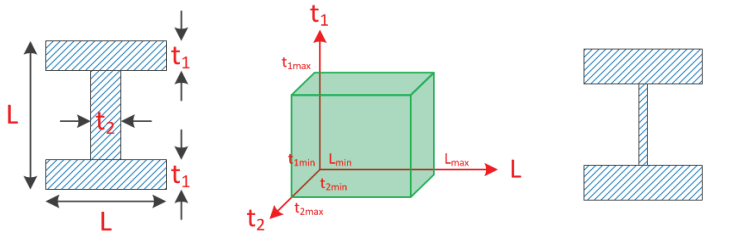


FIGURE 4.2: Instance representation in 3D space



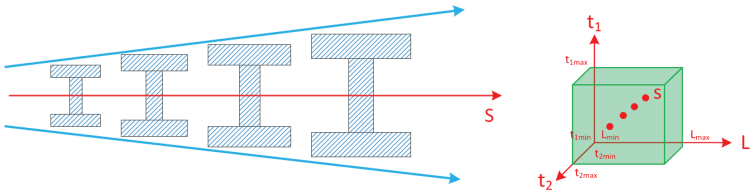
(a) I-shape section (b) 3D design space (c) Impractical case

FIGURE 4.3: Illustration of 3-parameter continuous design space

all necessary attributes, but in fact only two of them are independent, and the third one can be derived through the equation $G = E/(1+2\nu)$. As a result, some of the attributes may be redundant and need to be eliminated, but the difficulty arises when the correlations or dependencies are implicit, or are hidden in the data, making the optimization problem difficult to simplify by the designers.

Another consideration is that the admissible configurations represented by discrete points in the high-dimensional design space may follow some patterns that may be characterized by an even lower number of variables. For example, we consider an I-shape beam cross-section parameterized by three variables: t_1, t_2 and L (Fig. 4.3(a)). Continuous space corresponds to a cuboid in R^3 (Fig. 4.3(b)) allowing for any combination of the three variables. In continuous formulation, without additional constraints, the extreme design (t_{1max}, t_{2min}, L_{min}) is admissible resulting in an impractical configuration (Fig. 4.3(c)) with an unreasonable ratio of t_1 and t_2. On another hand, in practical cases, manufacturers provide a catalog of cross-section shapes to choose from. For instance, this catalog may allow only for a discrete subset of shapes which may be parameterized by a single homothetic transformation depending on only one latent (hidden) parameter (Fig. 4.4(a) and (b)). It means the intrinsic dimensionality of the catalog is one and it may be represented by a 1-D line pattern in 3-D space Fig. 4.4(b).

In the present work, we introduce tools for automatic discovery of such underlying patterns and for elimination the redundant dimensionality.



(a) Given catalog (b) the 1D pattern

FIGURE 4.4: Illustration of using one parameter to describe a series of cross-sections

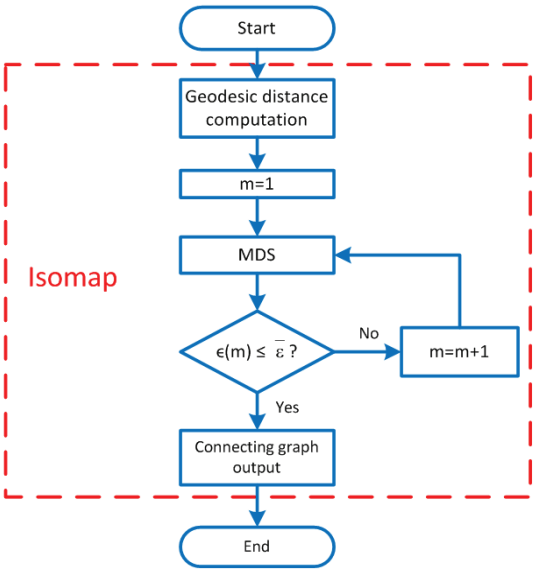


FIGURE 4.5: The flowchart of dimensionality reduction

4.3.1 Dimensionality reduction

The dimensionality reduction can be represented as a mapping from a higher dimensional to a lower dimensional space:

$$\boldsymbol{R}^{\mathrm{M}} \Rightarrow \boldsymbol{R}^{\mathrm{m}}, m \leq M \tag{4.2}$$

In this work, since the multi-dimensional discrete representation of categorical variables may be nonlinear, we choose Isomap technique (Tenenbaum, De Silva, and Langford, 2000) which introduces geodesic distance measures to classical MDS to reduce the redundant dimensionality. Fig. 4.5 shows the whole process of dimensionality reduction, and each step in the flowchart will be detailed in the further subsections.

4.3.2 Multi-dimensional scaling

The MDS algorithm (Martin and Eroglu, 1993) gives the reduced-order representation in following steps. Given the distance matrix:

$$\begin{aligned} &\mathbf{D}^{(\mathbf{X})}_{\mathrm{n}\times\mathrm{n}}, \mathbf{D}^{(\mathbf{X})}(\mathrm{i},\mathrm{j}) = ||(\mathbf{x}^{\mathrm{i}}, \mathbf{x}^{\mathrm{j}})||_2, \\ &\mathrm{i},\mathrm{j} = 1, 2, \cdots, \mathrm{n}, \end{aligned} \tag{4.3}$$

where $||\cdot||_2$ is the 2-norm of a vector. We aim at finding the new coordinates of $\mathbf{X}$ in lower dimensional space:

$$\begin{aligned} &\mathbf{Y}_{m\times n} = [\mathbf{y}^1, \mathbf{y}^2, \cdots, \mathbf{y}^n], \\ &\mathbf{y}^i = ({}^1b^i, {}^2b^i, \cdots, {}^mb^i)^T, \ \ i = 1, 2, \cdots, n, \\ &\sum_{i=1}^{n} {}^lb^i = 0, \ \ l = 1, 2, \cdots, m. \end{aligned} \tag{4.4}$$

In Eq. (4.4), a centering operation is carried on.
In order to operate on centred data, we define the transformation matrix $\mathbf{P}$:

$$\mathbf{P} = \mathbf{I} - \frac{1}{n}\mathbf{e}\mathbf{e}^T, \mathbf{e}^T = (1, 1, 1, \cdots, 1)_{1\times n}, \tag{4.5}$$

then we obtain the Gram matrix $\mathbf{G}$:

$$\mathbf{G} = -\frac{1}{2}\mathbf{P}(\mathbf{D}^{(\mathbf{X})})^2\mathbf{P}^T \tag{4.6}$$

Because $\mathbf{G}$ is symmetric and positive definite, the eigenvalue decomposition yields:

$$\mathbf{G} = \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^T, \tag{4.7}$$

where

$$\begin{aligned} &\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \lambda_2, \cdots, \lambda_M), \\ &\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_M \end{aligned} \tag{4.8}$$

and $\mathbf{\Phi}$ is the matrix whose columns contain eigenvectors of $\mathbf{G}$ and λ_i corresponds to the i-th largest eigenvalue.
We select the m largest eigenvalues and define:

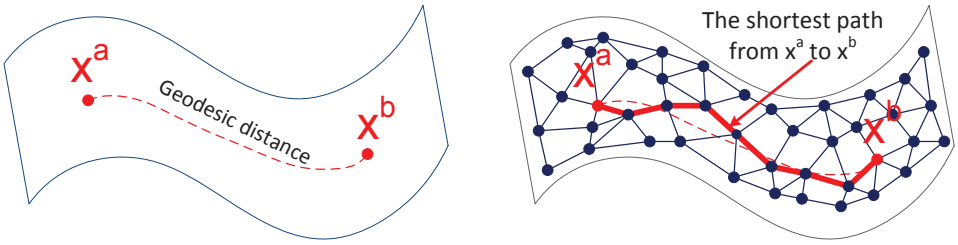
$$\mathbf{\Lambda}^{\frac{1}{2}} = [\mathrm{diag}(\sqrt{\lambda_1}, \sqrt{\lambda_2}, \cdots, \sqrt{\lambda_m}), \mathbf{0}_{m\times(M-m)}] \tag{4.9}$$

Finally we obtain the reduced-order coordinate matrix $\mathbf{Y}$:

$$\mathbf{Y} = [\mathbf{y}^1, \mathbf{y}^2, \cdots, \mathbf{y}^n] = \mathbf{\Lambda}^{\frac{1}{2}}\mathbf{\Phi}^T. \tag{4.10}$$

4.3.3 Isomap

Isomap (Tenenbaum, De Silva, and Langford, 2000) is built on the classical MDS framework but the main difference between Isomap and MDS is that Isomap replaces Euclidean distance by geodesic distance providing the ability to learn nonlinear data.
The fundamental hypothesis is that the data is sampled from a low dimensional manifold embedded in a high dimensional space. The geodesic distance is the shortest distance from point A to point B along the manifold (Fig. 4.6(a)). The Isomap approximates the geodesic distance in two steps.
In the first step, Isomap defines the nearest neighbour points for all the sample points by the ε-mean or the K-mean algorithm(Tenenbaum, De Silva, and Langford, 2000). Then, for every pair of neighbour points, their geodesic distance is approximated by the Euclidean distance; for other points, their geodesic distance is approximated by summing up the distances along the shortest path obtained by Dijkstra algorithm (Dijkstra, 1959) in the graph (Fig. 4.6(b)).



(a) Geodesic distance (b) The shortest path

FIGURE 4.6: Illustration of the geodesic distance

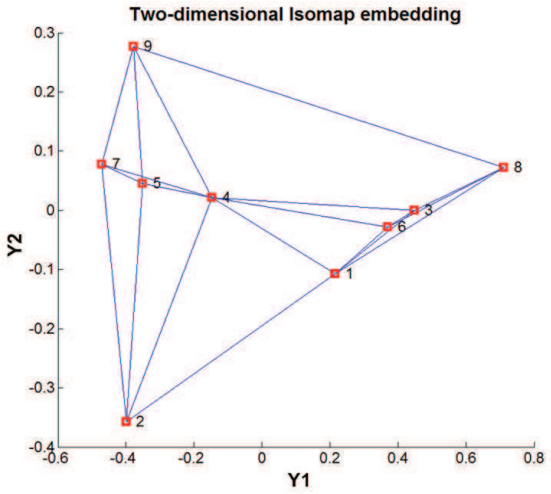


FIGURE 4.7: An example of dimensionality reduction of Fig. 4.2 using Isomap

In the second step, the MDS is executed with the geodesic distance matrix to obtain the reduced-order representation. Fig. 4.7 shows that Isomap reduces the dimensionality of the design space shown in Fig. 4.2 from a dimension of 3 to 2. However, Isomap requires an estimation of the target dimensionality m.

4.3.4 Intrinsic dimensionality estimation

The intrinsic dimensionality of the data in R^{M} is the lowest value of m allowing to represent data in a vector space of dimension m without (or with controlled) information loss. In this subsection, the criterion used to detect the intrinsic dimensionality is brought forward. As our goal is to preserve the geodesic distances, we propose to evaluate the correlation of the distance matrix by:

$$C(m) = \frac{Cov(\mathbf{Dv}_{\mathrm{X}}, \mathbf{Dv}_{\mathrm{Y}}^{m})}{\sqrt{Cov(\mathbf{Dv}_{\mathrm{X}}, \mathbf{Dv}_{\mathrm{X}})Cov(\mathbf{Dv}_{\mathrm{Y}}^{m}, \mathbf{Dv}_{\mathrm{Y}}^{m})}} \tag{4.11}$$

where $\mathbf{Dv}_{\mathrm{X}}$ is a vector extracted from the initial geodesic distance matrix, and $\mathbf{Dv}_{\mathrm{Y}}^{m}$ indicates the vector converted from the distance matrix in dimension m after the mapping. $Cov(\mathbf{u}, \mathbf{w})$ is the correlation coefficient of the two vectors $\mathbf{u}$ and $\mathbf{w}$.
Then, a positive threshold $\bar{\varepsilon}$ within the range [0,1] is set to demarcate the residual

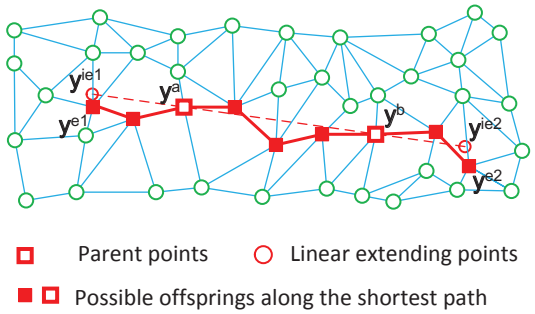


FIGURE 4.8: The graph-based crossover operator

variance. Finally, the intrinsic dimensionality is the smallest m such that:

$$\varepsilon(m) = \frac{|C(m) - C(M)|}{C(M)} \leq \bar{\varepsilon}. \tag{4.12}$$

After the construction of reduced-order discrete design space, the evolutionary algorithms can be applied.

4.4 Graph-based crossover and mutation operators

In this work, evolutionary algorithms (Coello, Van Veldhuizen, and Lamont, 2002) are implemented in the reduced-order space with crossover and mutation operators specifically designed for producing only admissible designs.

4.4.1 Graph-based crossover operator

We construct a custom-built crossover operator (Herrera, Lozano, and Verdegay, 1998) in the discrete design space operating on the graph. The steps of crossover are shown in Fig. 4.8.
Firstly, we select two parents $\mathbf{y}^a$ and $\mathbf{y}^b$. A linear extension through $\mathbf{y}^a$ and $\mathbf{y}^b$ on the graph is carried out

$$\begin{aligned} \mathbf{y}^{ie1} &= -0.5(\mathbf{y}^b - \mathbf{y}^a) + \mathbf{y}^a \\ \mathbf{y}^{ie2} &= 1.5(\mathbf{y}^b - \mathbf{y}^a) + \mathbf{y}^a \end{aligned} \tag{4.13}$$

Generally, the two extended endpoints $\mathbf{y}^{ie1}$ and $\mathbf{y}^{ie2}$ are not admissible, so we replace them by their nearest points $\mathbf{y}^{e1}$ and $\mathbf{y}^{e2}$.
Next we build the shortest path (Dijkstra, 1959) from $\mathbf{y}^{e1}$ to $\mathbf{y}^{e2}$ passing through $\mathbf{y}^a$ and $\mathbf{y}^b$, forming the crossover path $\mathscr{P}$. Finally, the offspring is randomly selected along $\mathscr{P}$.
This procedure guarantees producing admissible offsprings in the discrete design space.

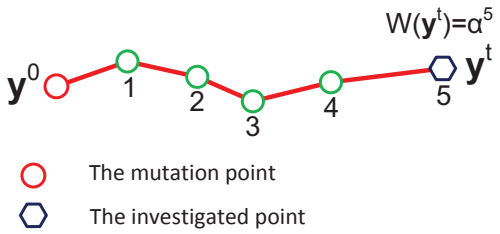


FIGURE 4.9: The graph-based mutation operator

4.4.2 Graph-based mutation operator

The mutation operator is intended to keep population diversity and helps protecting the optimization process from local solutions. This graph-based mutation operator consists of three main steps, including the neighbour layer recognition, mutation probability distribution and roulette wheel selection.
The neighbour layer recognition works as follows: given a mutation point $\mathbf{y}^0$, we can classify all the other points as different layers of neighbours with respect to $\mathbf{y}^0$. As shown in Fig. 4.9, for any point $\mathbf{y}^t$ different from $\mathbf{y}^0$, we can obtain the shortest path from $\mathbf{y}^0$ to $\mathbf{y}^t$ by Dijkstra algorithm. After that, the number of points l_t along the shortest path ($\mathbf{y}^0$ excluded) is counted and l_t gives the corresponding layer number; then $\mathbf{y}^t$ is defined as the l_t-th layer of neighbourhood point of $\mathbf{y}^0$. Noting that $\mathbf{y}^t$ is arbitrarily chosen, we can perform this step for all the points ($\mathbf{y}^0$ excluded) in the graph because we consider all those points as mutation candidates. Finally the layer numbers of all the points are calculated.
In the mutation probability distribution step, we calculate the mutation possibility for all neighbours in the graph. Firstly we suppose the point $\mathbf{y}^t$ belongs to the l_t-th layer of neighbour points of $\mathbf{y}^0$, then we give $\mathbf{y}^t$ an exponential mutation probability weight, as shown in Fig 4.9:

$$W(\mathbf{y}^t) = \alpha^{l_t}, \alpha \in (0, 1] \tag{4.14}$$

α is called the probability attenuation factor for different layers of neighbours. If α equals 1, it means every point in the graph shares the same mutation probability weight; if α is smaller than 1, it means the mutation probability weight is attenuated layer by layer by the percentage of α. It also provides us the flexibility to control the global distribution of mutation probability weights: if α is closer to 1, the operator tends to take a global mutation; while if α is closer to 0, a local mutation is more likely to happen. After we obtain all the mutation probability weights, we can assign the mutation probability to each point in the same layer by:

$$Prob(\mathbf{y}^t) = \frac{W(\mathbf{y}^t)}{\sum_{i=1}^{n} W(\mathbf{y}^i)}, \quad \mathbf{y}^i \neq \mathbf{y}^0. \tag{4.15}$$

In the third step, we use a standard roulette rule (Houck, Joines, and Kay, 1995) to select the mutation result.
This graph-based operator ensures admissible mutation results. Thus the rounding-off process in the real mutation for the discrete design space can be avoided.

4.5 Numerical tests

In this section, three numerical tests of various complexity are carried out. The first two concern optimization of a dome structure with a single categorical variable and with seven categorical variables; the third one is an optimal design of a six-story rigid frame structure with eight categorical variables.

4.5.1 The dome structure optimization with single categorical variable

In order to show how the discrete manifold-learning approach works, we consider a dome (Fig. 4.10) design problem (Csébfalvi, 2013). The structure consists of 120 bars, and is under vertical and gravity loads, with its base constrained to the ground. To obtain a better performance, we need to choose the cross-section types for all the bars, aiming at minimizing both the mass and the maximal ratio of axial loads and local critical buckling loads for each bar. Note, that the local buckling results only from compression load, so the critical buckling loads are treated as negative here. We assign all the 120 bars with the same cross-section chosen among 54 availabilities listed in Appendix A. Since we need to calculate the mass and the lateral buckling loads along local y and z directions, the cross-section areas, the area moments of inertia about local y and z-axis are the necessary attributes. The material properties are: Young's modulus $E = 2.1e11Pa$, Poisson's ratio $\nu = 0.3$ and the density $\rho = 7850kg/m^3$. The size of the initial population is 12. The crossover probability is 0.9, and the mutation probability is 0.03. The optimization problem is stated as follows:

$$\begin{aligned} \text{min.:} \quad & (\text{mass}(\mathbf{x}), \max(\frac{f_1}{f_{\text{cr1y}}}, \frac{f_1}{f_{\text{cr1z}}}, \frac{f_2}{f_{\text{cr2y}}}, \cdots, \frac{f_{120}}{f_{\text{cr120z}}})); \\ \text{s. t.:} \quad & \mathbf{Ku} = \mathbf{P}; \\ & \mathbf{x} \in \{\mathbf{x}^1, \mathbf{x}^2, \cdots, \mathbf{x}^{54}\}; \\ & \mathbf{x}_i^j = (A_i^j, I_{yi}^j, I_{zi}^j)^T, \ \ j = 1, 2, \cdots, 54. \end{aligned} \tag{4.16}$$

The $\bar{\varepsilon}$ used to detect the intrinsic dimension is set as 0.01. In this problem, the intrinsic dimension obtained equals 2, as listed in Tab. 4.2. Fig. 4.11(a) gives the design space with 54 admissible instances, while Fig. 4.11(b) shows the 54 instances in the embedded 2D space. When the Pareto front obtained remains unchanged in the next 2 generations, the searching process are regarded as converged. The maximum generation number is set as 50. In this test, the algorithm stops at generation 6 because the solutions in generation 5 and 6 are the same as those of generation 4. Fig. 4.12(a-d) shows the Pareto set designs in the design space, while the performance of the Pareto solutions is displayed in Fig. 4.13.

TABLE 4.2: Correlation coefficients and ε versus dimension m

m	1	2	3
C(m)	0.95030	0.96653	0.96737
$\varepsilon(m)$	0.0176	0.00087	

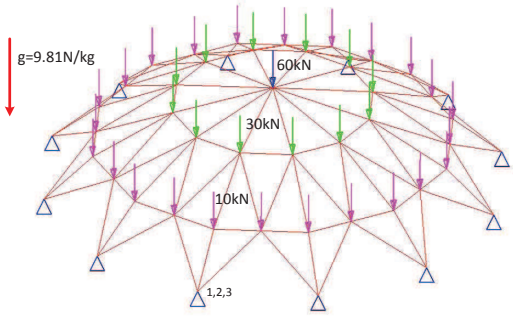
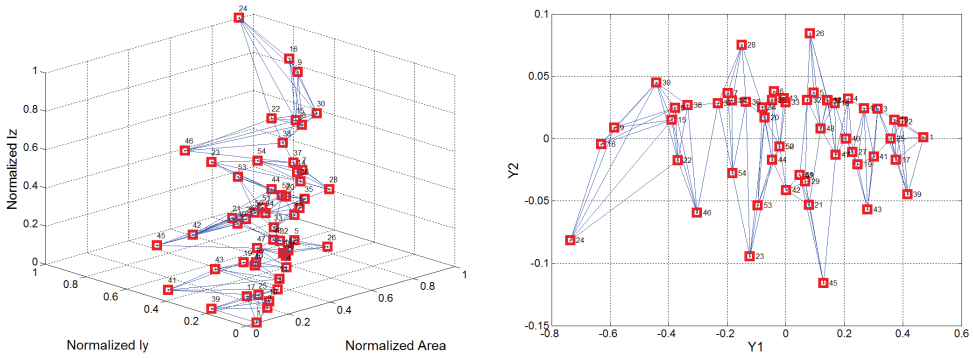


FIGURE 4.10: The dome structure with one categorical variable



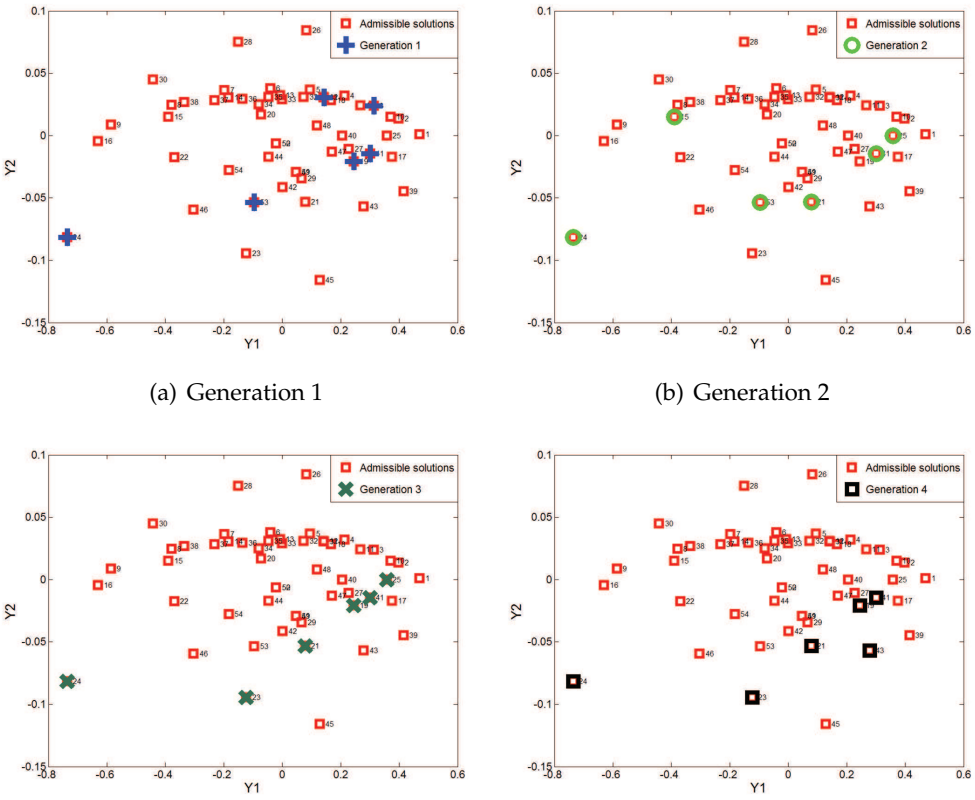
(a) The initial 3D design space

(b) The 2D reduced-order space

FIGURE 4.11: The mapping from 3D design space to 2D reduced-order space

4.5.2 The dome structure optimization with seven categorical variables

We consider the same dome structure with all the bars divided into seven groups, as shown in Fig. 4.14. Each group of bars shares one categorical variable, indicating that their cross-sections are the same. The materials for each bar, the loads and boundary conditions remain the same as in the first example. The size of the initial population is 200, and the number of generation is limited to 40. The optimization problem is stated as:



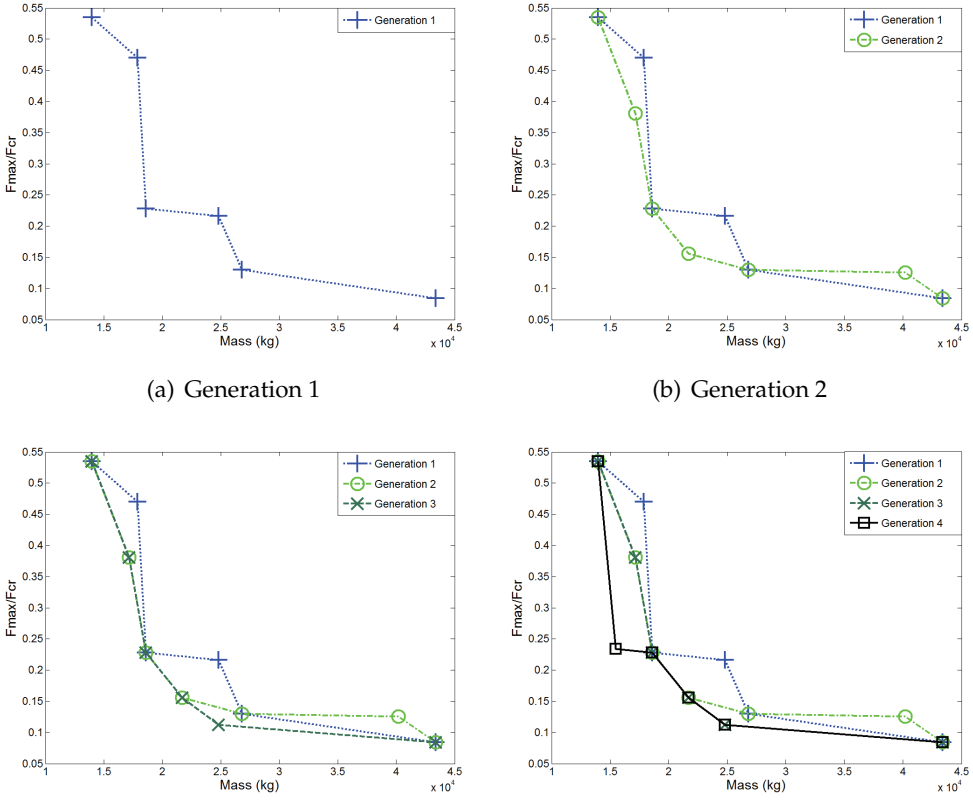
(a) Generation 1

(b) Generation 2

(c) Generation 3

(d) Generation 4

FIGURE 4.12: The design updating of Pareto sets in reduced-order space



(a) Generation 1

(b) Generation 2

(c) Generation 3

(d) Generation 4

FIGURE 4.13: The improving performance of Pareto sets in objective space

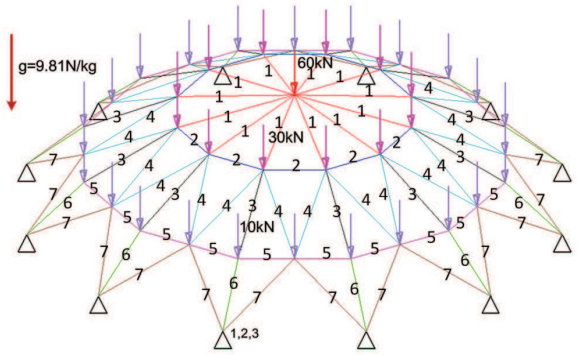


FIGURE 4.14: 7 groups of bars: each group shares the same cross-section

$$\begin{aligned} \text{min.:} \quad & (\text{mass}(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_7), \max(\frac{f_1}{f_{\text{cr1y}}}, \frac{f_1}{f_{\text{cr1z}}}, \frac{f_2}{f_{\text{cr2y}}}, \cdots, \frac{f_{120}}{f_{\text{cr120z}}})); \\ \text{s. t.:} \quad & \mathbf{Ku} = \mathbf{P}; \\ & \mathbf{x}_i \in \{\mathbf{x}_i^1, \mathbf{x}_i^2, \cdots, \mathbf{x}_i^{54}\}, \ i = 1, 2, \cdots, 7; \\ & \mathbf{x}_i^j = (A_i^j, I_{y_i}^j, I_{z_i}^j)^T, \ j = 1, 2, \cdots, 54. \end{aligned} \tag{4.17}$$

Fig. 4.15 shows the evolutionary optimization history of the populations, together with the Pareto set obtained in generation 40. To make a comparison, we use the sequential quadratic programming (SQP) method to solve the problem considering all the variables as continuous. We use the weighted sum of the two objectives, and then we find a solution in the continuous design domain which dominates all the other solutions displayed in Fig 4.16. Apparently, this solution is an inadmissible one. We round it off to the nearest admissible design in the design space (Fig. 4.17), then we evaluate the rounded-off design. Fig. 4.16 shows the continuous approach provides a single solution which is not admissible regarding the discrete design space. Rounding-up this solution to the closest feasible design still yields a single solution emphasizing on structural buckling performance, which is slightly dominated by one of the designs obtained by the proposed approach. We also illustrate their differences in the design space, as shown in Fig. 4.17.
Remark that in this test, the total mass is exclusively related to the cross-section areas, while the buckling safety factors only rely on the area moments of inertia about y and z directions. That means if we consider the design space as continuous, we can minimize both objectives at the same time without any compromise. That's why in this test we can obtain one single "perfect" solution which dominates all the others. But when we take a finite set of cross-section areas and the area moments of inertia into account, there may be no "perfect" solution anymore, and what we can get is a set of best compromise designs represented on the Pareto front.

4.5.3 Dam structure design

In this example. the dam structure is composed by three groups of bars, and different group of bars are divided by different colours, as shown in Fig. 4.18. The dame base is fixed on the ground and the nodes of the structure on the right are suffering four external forces 10000N pointing to the left. The available cross-sections are also

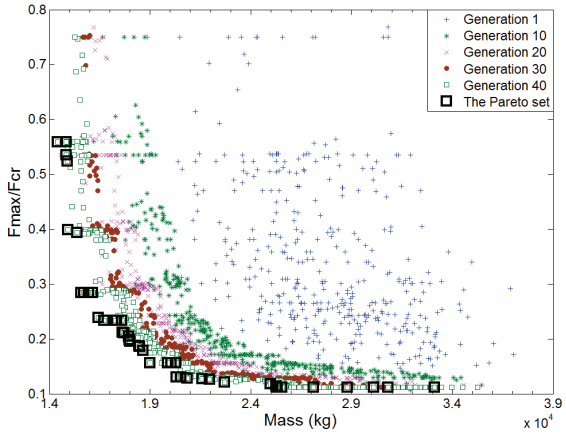


FIGURE 4.15: The evolutionary design history in objective space and the final Pareto set

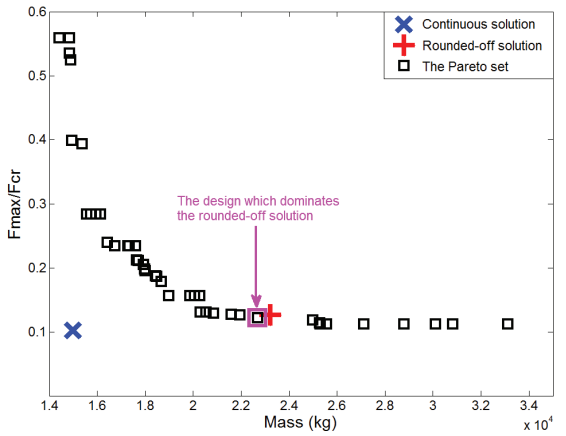


FIGURE 4.16: The result comparison between the discrete manifold-learning approach and the rounded-off solution

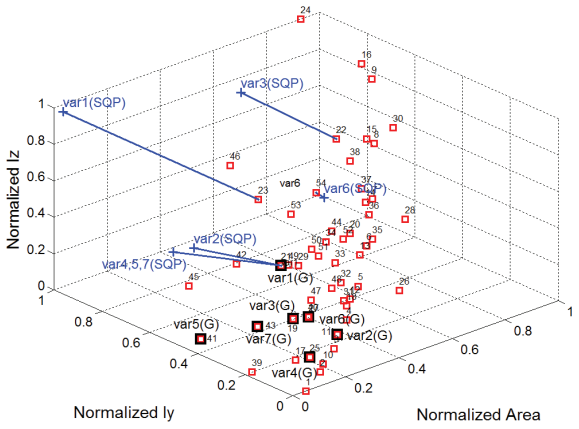


FIGURE 4.17: Designs comparison in design space: the better design (var1-7(G)) by the discrete manifold-learning approach and the rounded-off process from SQP solution (var1-7(SQP)) to nearest admissible instances

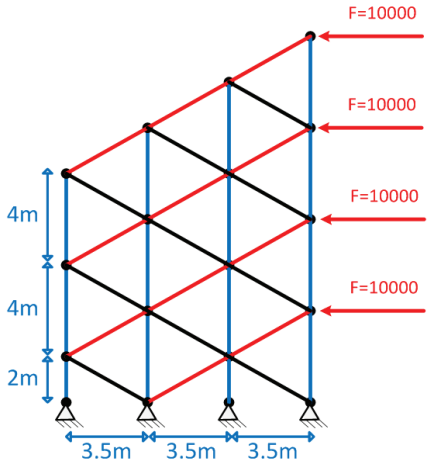


FIGURE 4.18: The dame-like structure

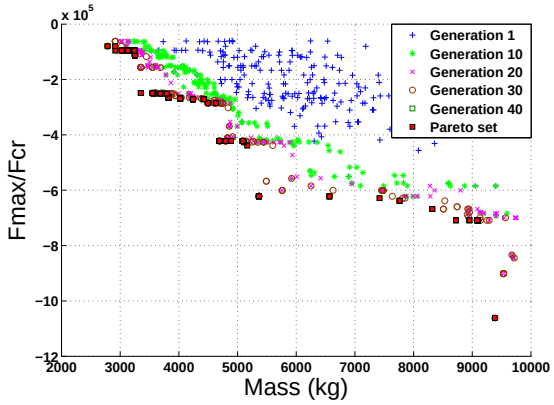


FIGURE 4.19: The evolutionary design history in objective space and the final Pareto set

shown in previous example. The material properties for all the bars are: Young's modulus E $= 2.1e11$Pa and the density $\rho = 7850\text{kg}/m^3$. The optimization statement is written as follows:

$$\begin{aligned} \text{min.:} \quad & (\text{mass}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3), \max(f_{cr1y} - f_1, f_{cr1z} - f_1, f_{cr2y} - f_2, \cdots, f_{cr35z} - f_{35})); \\ \text{s. t.:} \quad & \mathbf{Ku} = \mathbf{P}; \\ & \mathbf{x}_i \in \{\mathbf{x}_i^1, \mathbf{x}_i^2, \cdots, \mathbf{x}_i^{54}\}, \ \ i = 1, 2, 3; \\ & \mathbf{x}_i^j = (A_i^j, I_{yi}^j, I_{zi}^j)^T, \ \ j = 1, 2, \cdots, 54. \end{aligned} \tag{4.18}$$

We display the evolutionary design history of the generations 1, 10, 20, 30 and 40 as in Fig. 4.19, together with the final Pareto set. In 4.19, we can find that the final Pareto set mainly distribute in four clusters, and each cluster of Pareto points have similar performance on the buckling, while their masses have significant differences. It reminds us that we can choose the Pareto points which have the minimum mass as our designs since they are more competitive than the other Pareto points.

4.5.4 Six-story rigid frame structure design

The goal of the fourth example is to compare the discrete manifold-learning approach with the simplex (Filomeno Coelho, 2014). This example (Coelho et al., 2015) is a typical beam section-type selection problem. The rigid frame structure consists of a total of 63 beams which are divided into eight groups (Fig. 4.20). Each group of beams shares the same cross-section chosen from 54 cross-section instances. All the cross-section types are listed in Appendix B, together with their physical attributes: areas, area moments of inertia about y-axis, area moments of inertia about z-axis and torsion moments of inertia about x-axis. We set the intrinsic dimension detection parameter $\bar{\varepsilon}$ as 0.01, and the intrinsic dimension m=3 (Tab. 4.3). The reduced-order representation of the 54 instances in 3D space is shown in Fig. 4.21. Different beams are manufactured with the same material as in the first example. The lengths of horizontal beams are 7.315m, while the vertical beams are 3.658m long.
The rigid frame structure is subject to three kinds of loads: 1) The gravity load on the floor (19.16kPa); 2) The dead load of the beams; 3) The lateral load caused by the wind (110kN).
The structural analysis of the rigid frame structure is performed employing a finite element program (Ferreira, 2008). The goal is to minimize both the mass and the compliance of the frame. We state the optimization problem as:

$$\begin{aligned} \text{min.:} \quad & (m(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_8), \mathrm{Se} = 0.5\mathbf{u}^{\mathrm{T}}\mathbf{K}\mathbf{u}); \\ \text{s. t.:} \quad & \mathbf{K}\mathbf{u} = \mathbf{F}; \\ & \mathbf{x}_i \in \{\mathbf{x}_i^1, \mathbf{x}_i^2, \cdots, \mathbf{x}_i^{54}\}, \ \ i = 1, 2, \cdots, 8; \\ & \mathbf{x}_i^j = (A_i^j, Iy_i^j, Iz_i^j, Jx_i^j)^{\mathrm{T}}, \ \ j = 1, 2, \cdots, 54. \end{aligned} \tag{4.19}$$

We set the population size as 100, and the number of generation is limited to 40.
Fig. 4.22 shows the evolutionary design history of different generations. We also compare the Pareto set solutions with the results obtained by rounding off continuous solutions to nearest admissible solutions in design space (Fig. 4.23). Those solutions (+ symbols) clearly produce heavier weight and are outperformed by the categorical ones (square symbols).

TABLE 4.3: Correlation coefficients and ε versus dimension m

m	1	2	3	4
C(m)	0.7950	0.8355	0.8873	0.8944
$\varepsilon(m)$	0.1028	0.0608	0.0073	

Here we compare the proposed approach with simplex by testing the design problem of the six-story rigid frame structure. Due to the stochastic nature of both approaches, to avoid the effect of randomness, the optimization is run independently for 20 times with the number of generations limited to 60 and the population size of 100. All the optimization parameters are given as in the previous test.
We gather the Pareto fronts in generation 10, 20, 30, 40, 50 and 60 obtained by simplex and the proposed approach in the 20 runs and plot them together in Fig. 4.24(a-f). We can find that both of the Pareto sets obtained by the two approaches have

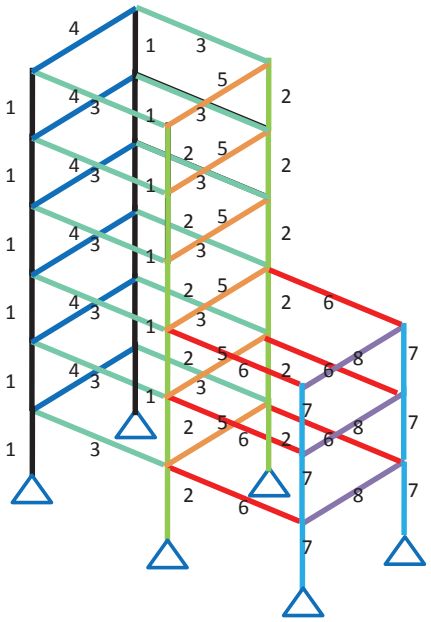


FIGURE 4.20: The 6-storey rigid frame structure

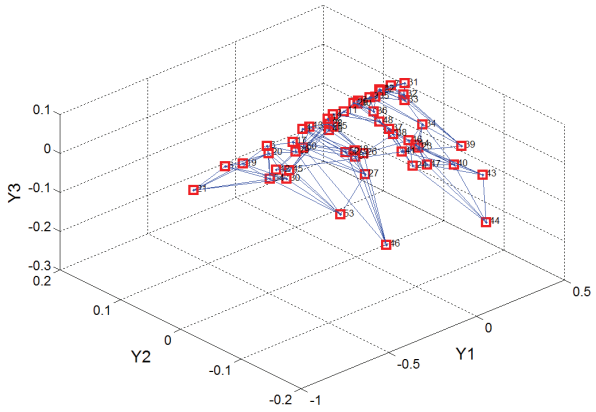


FIGURE 4.21: Design instances represented in reduced-order 3D space

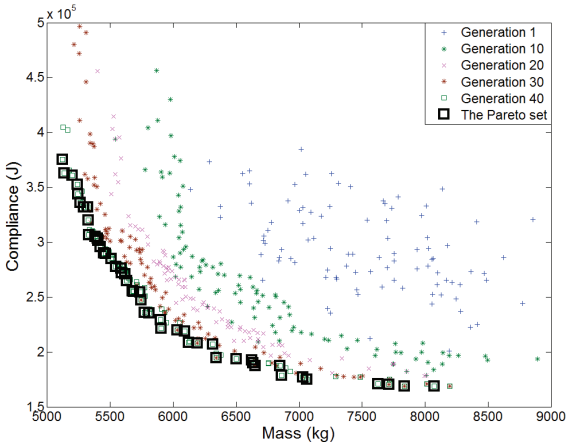


FIGURE 4.22: The evolution of designs in objective space and the final Pareto set

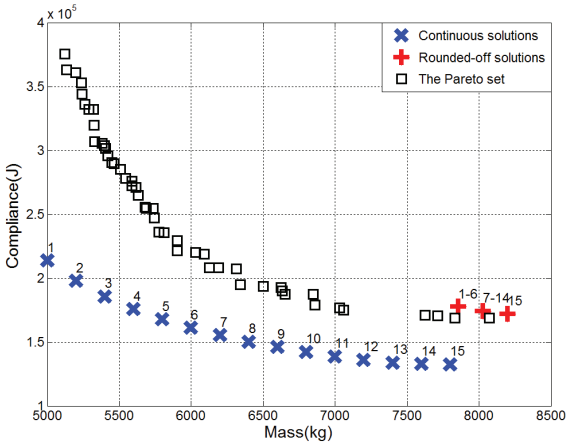
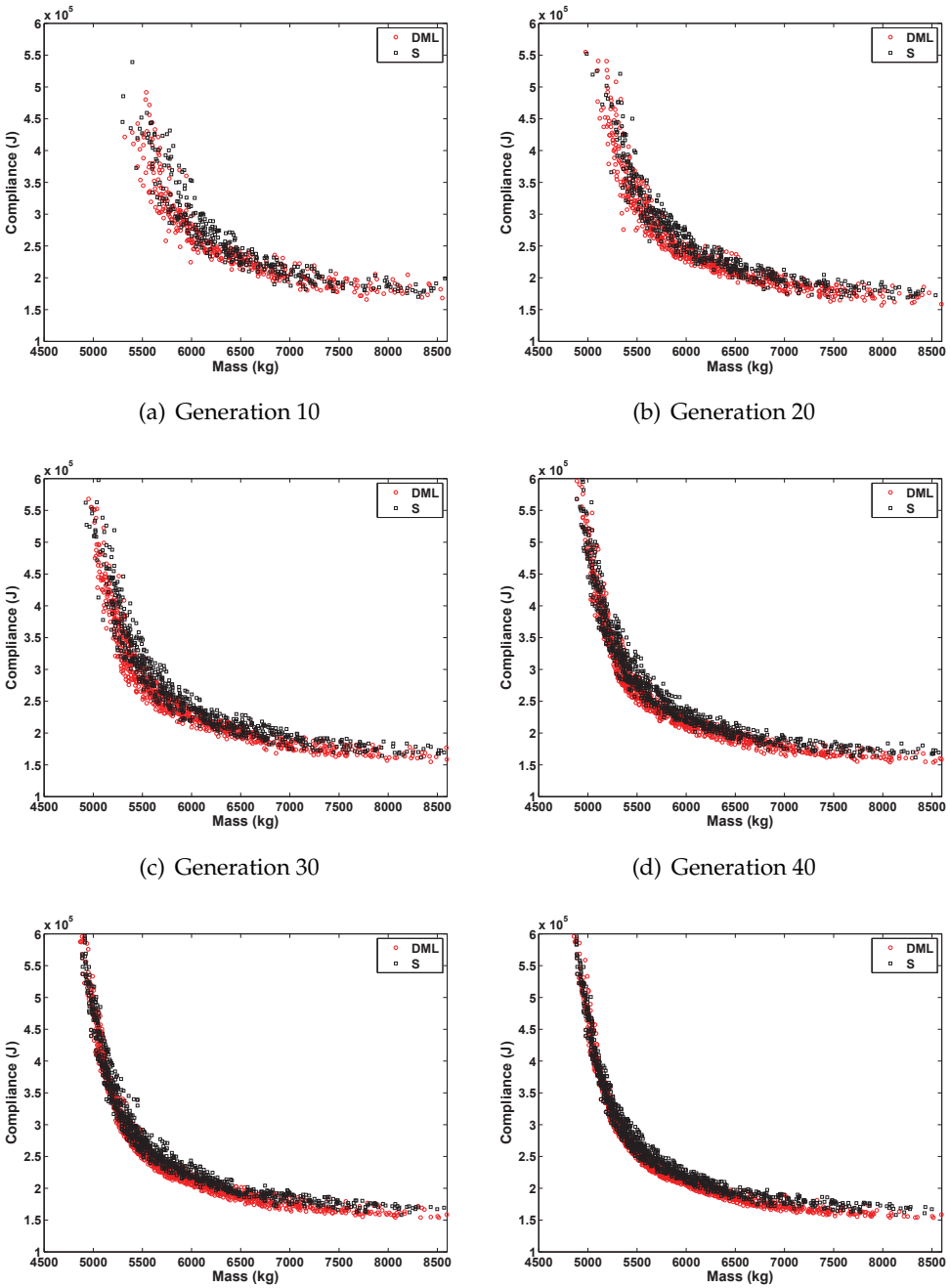


FIGURE 4.23: The result comparison between the discrete manifold-learning approach and the continuous solutions rounded off to the nearest admissible ones

experienced a clustering process to the ideal Pareto fronts, illustrating the convergence ability of the two approaches. In order to compare their performance, we fit them with six order polynomials, respectively, because six order polynomials suits the Pareto sets well and bring less fitting errors. We display the corresponding fitting curves and the standard deviations of the fitting errors of each selected generation for the two approaches in Fig. 4.25. The six figures show that in most cases, the proposed discrete manifold learning approach can obtain better Pareto sets than simplex. From generation 10 to 60, the difference of the two fitting curves mainly occurs in the medium-mass and heavy-mass segments, namely from 5300kg to 8600kg; while the light-mass segment (below 5300kg) shows no apparent differences.
In order to make a meticulous comparison, the standard deviations of the fitting errors of each generation are displayed in Fig. 4.26. Fig. 4.26 shows that: 1) From generation 10 to 30, the standard deviations of the fitting errors obtained by the two approaches are normally in the light-mass segment (below 5300kg), but keep low and steady in the medium-mass and heavy-mass segments (above 5300kg). After generation 40, the standard deviations of fitting errors in all segments are steady, which indicates the Pareto sets of 20 different runs remain stable to some extent.
2) As the generation number increases, the standard deviations of the fitting errors become smaller, namely decreasing from 22000J to 12000J (simplex) and 11000J (DML), which shows the Pareto sets are becoming more and more gathered and converged.
3) In generation 10, both of the two approaches have much similar standard deviations of fitting errors, In generation 20, simplex has better performance, but from generation 30 to 60, the proposed discrete manifold learning approach always yield smaller standard deviations of fitting errors, showing that the proposed approach still have advantages in optimization convergence.

4.6 Conclusions and prospects

The proposed approach allows for discovering implicit relationships between design variables. The dimensionality of a problem may be reduced by manifold learning techniques such as Isomap. The manifold learning approach yielded significant improvements when using a multi-dimensional catalog because it discovered the underlying 2D (3D in the third example) structure of the data also allowing for better comprehension of the design space, and ultimately permitting a faster discovery of Pareto set. In this work, we have proposed custom-built genetic operators including crossover and mutation. The obtained multi-objective algorithm allows for the treatment of categorical problems restraining by construction the optimal Pareto sets to admissible design points only.
Further work will consist in comparing our approach with a continuous, penalized formulation of categorical problems.



(a) Generation 10

(b) Generation 20

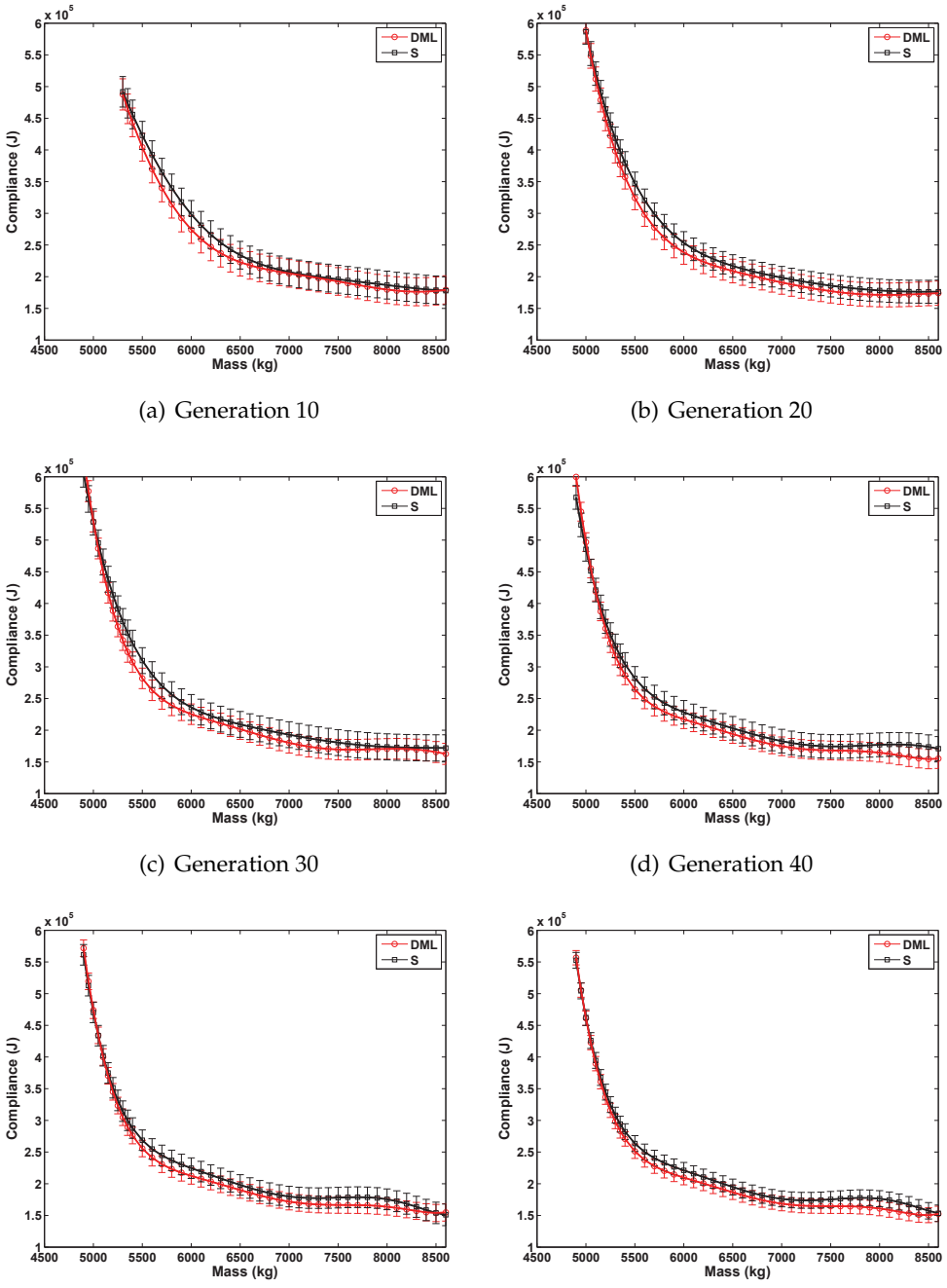
(c) Generation 30

(d) Generation 40

(e) Generation 50

(f) Generation 60

FIGURE 4.24: The Pareto sets of selected generations obtained by the discrete manifold learning approach (DML) and simplex (S) of 20 different runs



(a) Generation 10

(b) Generation 20

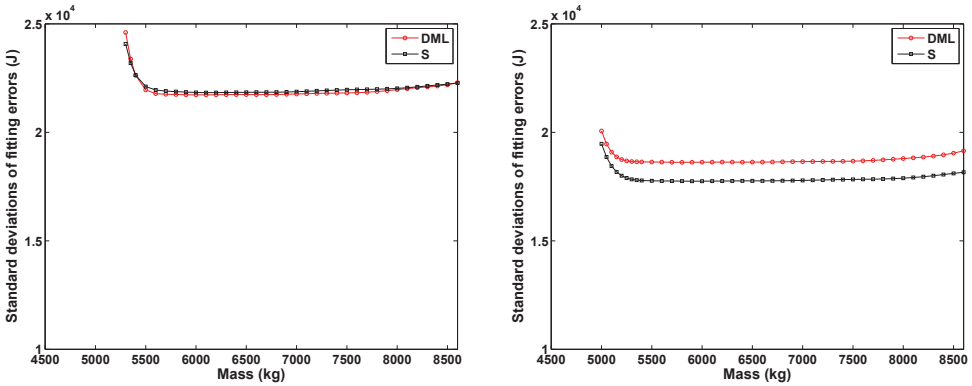
(c) Generation 30

(d) Generation 40

(e) Generation 50

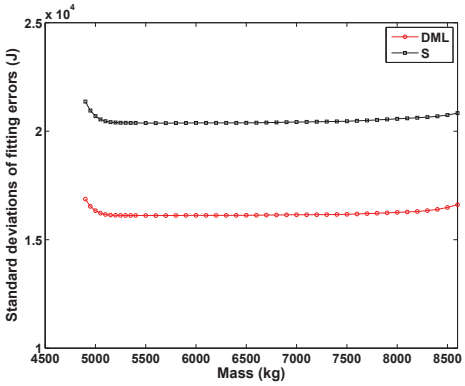
(f) Generation 60

FIGURE 4.25: The fitting curve together with fitting error estimation of the Pareto sets obtained by both approaches

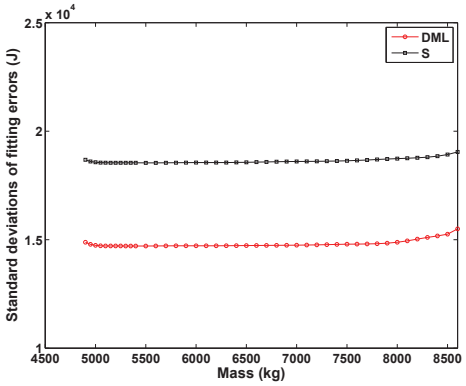


(a) Generation 10

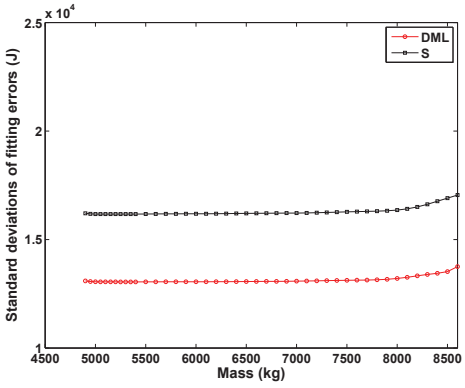
(b) Generation 20



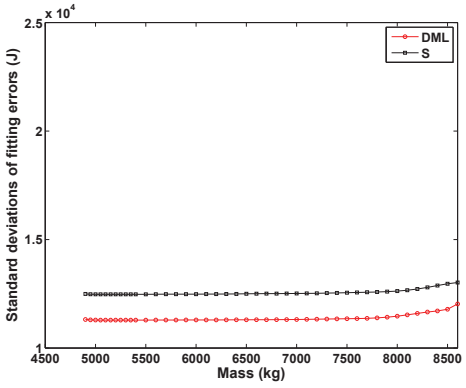
(c) Generation 30



(d) Generation 40



(e) Generation 50



(f) Generation 60

FIGURE 4.26: The comparison of standard deviations estimation of the fitting errors

Chapter 5

Categorical optimization: a two-stage search strategy

5.1 Background

In various optimization problems, design variables are generally classified as continuous, discrete (including integer) and categorical ones (Coelho et al., 2015). Among them, the categorical variable shows its particularity for it takes concrete instances within a category instead of real numbers. As a typical industrial example, a beam cross-section can only be taken from a finite set of fixed cross-sections. Those cross-sections differ both in shapes and sizes, thus it is tough to describe them by using a single parameter.

Classical methods (Herrera, Lozano, and Verdegay, 1998; Goldberg, 2006) to handle optimization problems with categorical variables commonly ignore the particularity: By encoding categorical variables with arbitrary real numbers or binary strings, the optimization is launched by evolutionary optimizers, for example the genetic algorithms. Those methods carry the advantages of possibly finding the global optima and no sensitivities needed, but different encodings may affect the optimization results.

Other works are proposed to treat categorical variables in a more specific manner. For example, when dealing with unordered categorical variables, simplex coding is applied to represent them, and this coding suits for evolutionary methods well (Coelho et al., 2015; Filomeno Coelho, 2014). In this coding, the distances between any two design instances are kept equal, thus forming a regular simplex in the space, whose dimensionality is always one less than the number of the instances, so if the number of instances is large, the dimensionality of the related space is also high. In previous chapters, we advocate using multi-dimensional discrete vectors to stand for categorical variables along with a specially tailored evolutionary optimizer.

In this chapter, benefiting from the multi-dimensional representation, the manifold learning technique is used to find the lower-dimensional manifold embedded in the original design space. Then based on the reduced-order representation, we approximate the manifold with a polynomial interpolation equations by using the Least Squares method. We optimize the constrained problem in the reduced-order space by gradient-based methods, for example the Method of Moving Asymptotes. In the last stage, we apply a discrete local search.

The main two linear manifold learning techniques are the Principal Component Analysis (PCA) (Jolliffe, 2002) and Multi-Dimensional Scaling (MDS) (Martin and Eroglu, 1993), respectively. By using the eigenvalue decomposition, PCA tries to preserve the most covariance information in the reduced order space, while MDS focuses on the preservation of Euclidean distances between samples points. But they do not have the ability to map a non-linear manifold to a reduced-order space, for

example the "Swiss Roll" manifold (Tenenbaum, De Silva, and Langford, 2000). The typical non-linear manifold learning techniques are Isomap (Tenenbaum, De Silva, and Langford, 2000; Balasubramanian and Schwartz, 2002), Locally linear Embedding (LLE) (Roweis and Saul, 2000; De Ridder et al., 2003) and Kernel Principal Component Analysis (KPCA)(Schölkopf, Smola, and Müller, 1998; Cao et al., 2003). LLE is developed to preserve the weights of local neighbours in a lower-dimensional space. One of the advantages is that the optimization problem to obtain the weights inside the algorithm is convex and the local minima can be avoided. The Isomap is a variant of MDS. The difference is that Isomap uses geodesic distances calculated by the Dijkstra algorithm (Dijkstra, 1959) instead of Euclidean distances used in MDS, so finally the geodesic distances can be preserved at the largest proportion. KPCA firstly maps data from a lower to higher dimensional space with a kernel function which makes the non-linear data become linear (Schölkopf, Smola, and Müller, 1998; Cao et al., 2003), then the Support Vector Machine (SVM) (Cortes and Vapnik, 1995; Vapnik, 2013) is applied to obtain the final solution. KPCA can be also regarded as a variant of PCA. Those linear and non-linear manifold learning techniques have already been used in identification problems (Meng et al., 2015), structural optimization (Raghavan et al., 2013) and visualization (Patwari, Hero III, and Pacholski, 2005).
We carry out the contents of this chapter as follows. In section 5.2, we state the categorical problem. In section 5.3, the continuous searching approach is illustrated. In section 5.4, the local neighbour search is introduced. The section 5.5, we execute numerical tests and make comparisons.

5.2 Categorical optimization statement

The values which a categorical variable can take are called instances, or levels (Banker and Morey, 1986). The instances are described by multi-dimensional discrete vectors, which can be represented as points in the multi-dimensional space. For the sake of generality, we present the constrained problem with e categorical variables:

$$\begin{aligned} \text{min.:} \quad & J(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_e); \\ \text{s. t.:} \quad & \mathbf{g}(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_e) \leq \mathbf{0}; \\ & \mathbf{x}_i \in \mathbf{X}_i = \{\mathbf{x}_i^1, \mathbf{x}_i^2, \cdots, \mathbf{x}_i^n\}, \ \ i = 1, 2, \cdots, e; \\ & \mathbf{x}_i^j = ({}^1a_i^j, {}^2a_i^j, \cdots, {}^Ma_i^j)^T, \ \ j = 1, 2, \cdots, n. \end{aligned} \tag{5.1}$$

In Eq. (5.1), J is the objective function, and $\mathbf{g}$ stands for the constraint functions. $\mathbf{x}_i$ denotes the i-th categorical variable, and it can take n instances as its value. $\mathbf{x}_i^j$ is the j-th instance of the i-th variable, and it is represented by a M-dimensional vector. ${}^1a_i^j$ is the l-th attribute for the j-th instance of variable i.

5.3 Continuous search stage

In this section, the manifold learning techniques, including PCA, MDS, Isomap, LLE and KPCA, are applied to reduce the dimensionality of original design space, and the lower-dimensional representation is obtained. Then the mapping relationship from the lower-dimensional representation to original representation is constructed.

Then the construction errors are estimated and compared. Finally the Method of Moving Asymptotes (MMA) (Svanberg, 1995) used in the chapter is briefly introduced.

5.3.1 Dimensionality reduction

As we have defined the design space for an any categorical variable $\mathbf{x}$ as:

$$\begin{aligned} &\mathbf{X} : [\mathbf{x}^1, \mathbf{x}^2, \cdots, \mathbf{x}^n], \\ &\mathbf{x}^j = ({}^1a^j, {}^2a^j, \cdots, {}^Ma^j)^T, \ \ j = 1, 2, \cdots, n. \end{aligned} \tag{5.2}$$

where $\mathbf{X}$ is a $M \times n$ matrix and $\mathbf{x}^j$ is the j-th instance. The dimensionality reduction can be illustrated as:

$$\begin{aligned} &\mathbf{X} \Rightarrow \mathbf{Y} : [\mathbf{y}^1, \mathbf{y}^2, \cdots, \mathbf{y}^n], \\ &\mathbf{y}^j = ({}^1b^j, {}^2b^j, \cdots, {}^mb^j)^T, \ \ j = 1, 2, \cdots, n. \end{aligned} \tag{5.3}$$

The corresponding operation to each instance is:

$$\mathbf{x}^j \Rightarrow \mathbf{y}^j \tag{5.4}$$

namely

$$\begin{aligned} &({}^1a^j, {}^2a^j, \cdots, {}^Ma^j)^T \Rightarrow ({}^1b^j, {}^2b^j, \cdots, {}^mb^j)^T, \\ &M > m, \ \ j = 1, 2, \cdots, n. \end{aligned} \tag{5.5}$$

The popular manifold learning techniques include PCA, MDS, Isomap, LLE and KPCA et. al. In this chapter, those manifold learning techniques mentioned above are applied and compared.
We need to note that, if the orders of magnitude along every dimension of the data are different, a normalization for each dimension is suggested before carrying out the dimensionality reduction. During the normalization, the data is normalized within the range between 0 and 1. While in the evaluation process, the normalized data will be remapped to the original data according to the lower and upper bounds.

5.3.2 The polynomial fitting and the fitting error estimation

In this section, we introduce a polynomial fitting to approximate the discrete manifold in the higher-dimensional space in a continuous manner:

$$(\mathbf{x}^{j^*})^T = \mathbf{p}^T(\mathbf{y}^j) \cdot \mathbf{D}, \ \ j = 1, 2, \cdots, n. \tag{5.6}$$

where $\mathbf{x}^{j^*}$ denotes the approximation of the instance $\mathbf{x}^j$, while $\mathbf{p}^T(\mathbf{y}^j)$ is the polynomial expression and $\mathbf{D}$ is the coefficient matrix for every item of the polynomial.
For example, if the polynomial order is 2, M equals 3 and m equals 2, then $\mathbf{p}^T(\mathbf{y}^j)$ can be written as:

$$\mathbf{p}^T(\mathbf{y}^j) = (1, {}^1b^j, {}^2b^j, ({}^1b^j)^2, ({}^2b^j)^2, {}^1b^j \cdot {}^2b^j) \tag{5.7}$$

and the corresponding coefficient matrix $\mathbf{D}$ is expressed as:

$$\mathbf{D} = \begin{bmatrix} d_{1,1} & d_{1,2} & d_{1,3} \\ d_{2,1} & d_{2,2} & d_{2,3} \\ \vdots & \vdots & \vdots \\ d_{10,1} & d_{6,2} & d_{6,3} \end{bmatrix}. \tag{5.8}$$

When the polynomial order is 3, the dimensionality M and m remain the same, then $\mathbf{p}^{\mathrm{T}}(\mathbf{y}^{\mathrm{j}})$ can be written as:

$$\mathbf{p}^{\mathrm{T}}(\mathbf{y}^{\mathrm{j}}) = (1, {}^{1}\mathrm{b}^{\mathrm{j}}, {}^{2}\mathrm{b}^{\mathrm{j}}, ({}^{1}\mathrm{b}^{\mathrm{j}})^{2}, ({}^{2}\mathrm{b}^{\mathrm{j}})^{2}, {}^{1}\mathrm{b}^{\mathrm{j}} \cdot {}^{2}\mathrm{b}^{\mathrm{j}}, ({}^{1}\mathrm{b}^{\mathrm{j}})^{3}, ({}^{2}\mathrm{b}^{\mathrm{j}})^{3}, ({}^{1}\mathrm{b}^{\mathrm{j}})^{2} \cdot {}^{2}\mathrm{b}^{\mathrm{j}}, {}^{1}\mathrm{b}^{\mathrm{j}} \cdot ({}^{2}\mathrm{b}^{\mathrm{j}})^{2}) \tag{5.9}$$

and $\mathbf{D}$ is expressed as:

$$\mathbf{D} = \begin{bmatrix} d_{1,1} & d_{1,2} & d_{1,3} \\ d_{2,1} & d_{2,2} & d_{2,3} \\ \vdots & \vdots & \vdots \\ d_{10,1} & d_{10,2} & d_{10,3} \end{bmatrix}. \tag{5.10}$$

In order to obtain matrix $\mathbf{D}$, we firstly define the fitting difference for instance j:

$$E_{\mathrm{j}} = \| \ (\mathbf{p}^{\mathrm{T}}(\mathbf{y}^{\mathrm{j}}) \cdot \mathbf{D})^{\mathrm{T}} - \mathbf{x}^{\mathrm{j}} \ \|_2 \tag{5.11}$$

where $\| \cdot \|_2$ indicates the 2 norm of a vector.
Then we minimize the overall approximation errors:

$$\text{min.:} \quad E(\mathbf{D}) = \sum_{\mathrm{j}=1}^{\mathrm{n}} E_{\mathrm{j}}^2. \tag{5.12}$$

By minimizing the criterion above, we can obtain the coefficient matrix $\mathbf{D}$.
After we get the $\mathbf{D}$, we can calculate the mean errors for all the instances:

$$E_{\text{mean}} = \frac{1}{n} E(\mathbf{D}). \tag{5.13}$$

And the maximum fitting error for all instances can be written as:

$$E_{\text{max}} = \max(E_{\mathrm{j}}^2, \ \mathrm{j} = 1, 2, \cdots, \mathrm{n}). \tag{5.14}$$

Both of the mean fitting error and the maximum fitting error are considered when we apply different manifold learning dimensionality reduction techniques into our

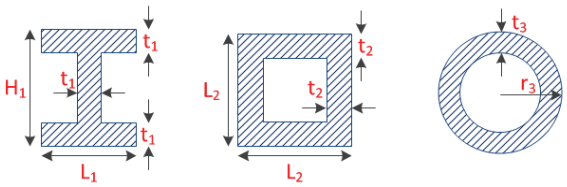


FIGURE 5.1: The three types of cross-sections

case.

5.3.3 A practical fitting example and comparison

Here we introduce the design instance catalog of all the bar cross-sections. In our case, we have three types of cross-sections: the I-shape, the hollow square shape and the hollow circle shape, as illustrated in Fig. 5.1. The geometrical parameters of all the shapes are listed in Tab. 5.1, respectively. Due to the varying geometrical parameters, there are totally 36 instances for the I-shape cross-section, 36 instances for the hollow square cross-section and 15 instances for the hollow circle cross-section.

TABLE 5.1: The available values of geometrical parameters

Parameter	Available values(m)
H_1	(0.05, 0.06, $\cdots$, 0.1)
L_1	(0.04, 0.05, $\cdots$, 0.09)
t_1	(0.005)
L_2	(0.04, 0.045, $\cdots$, 0.08)
t_2	(0.004, 0.005, $\cdots$, 0.007)
r_3	(0.04, 0.045, $\cdots$, 0.06)
t_3	(0.005, 0.006, 0.007)

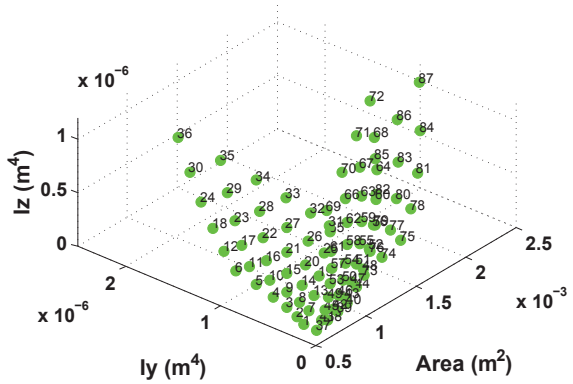


FIGURE 5.2: The physical properties of 87 cross-sections

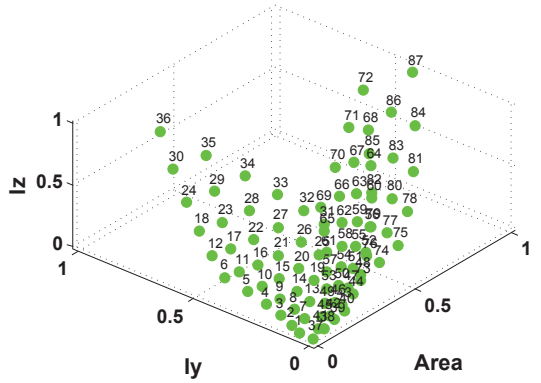


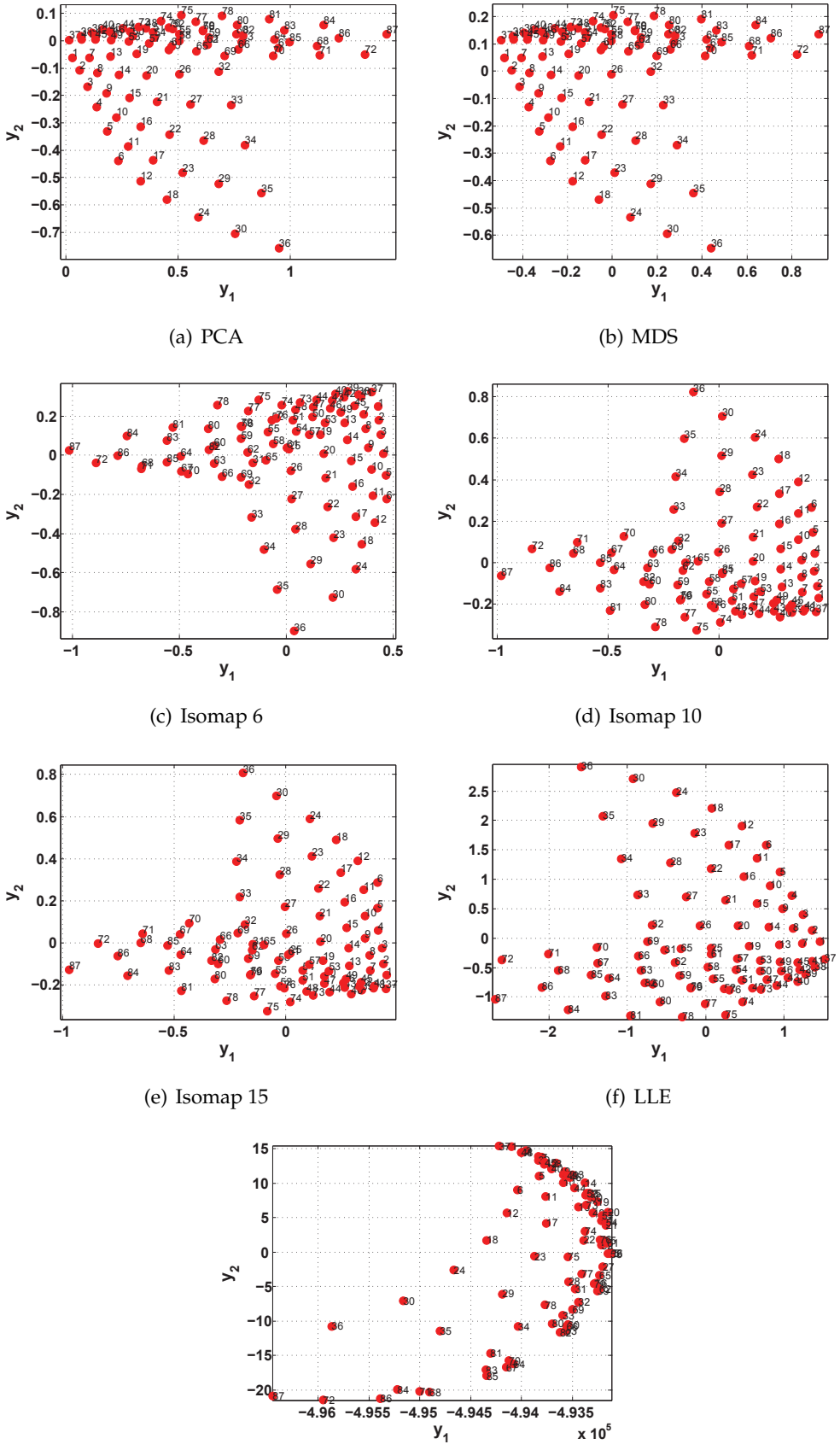
FIGURE 5.3: The normalized physical properties

The investigated physical properties are cross-section areas, area moments of inertia along y-axis and area moments of inertia along z-axis (x-axis is recognized as the normal direction of the area). We plot all 87 cross-sections in a 3-d space, as shown in Fig. 5.2, then we normalize them in Fig. 5.3.

We utilize PCA, MDS, Isomap, LLE and KPCA to reduce the dimensionality of the normalized design space from 3 to 2, respectively. Then we compare their fitting errors after applying the 2nd order polynomial interpolation. As the input parameters of Isomap contain the number of predefined neighbours, and they may influence the performance, so in this comparison, we set the number of neighbours to 6, 10 and 15, respectively. For LLE, the number of neighbours is set to 10. For KPCA, the kernel is chosen as the Gaussian Racial Based Function (GRBF). The reduced-order representations with different manifold learning techniques are illustrated in Fig. 5.4(a-g). Based on the 2D representation, the polynomial expressions are applied to yield the new 3D representations. The comparison between the original and the new 3D representations are illustrated in Fig. 5.6. We also investigate the fitting errors for each instance by different manifold learning method, and display them in Fig. 5.5. Note that the lines in all the figures indicate their average error level.

The mean fitting errors and the maximum fitting errors of each method are shown in Fig. 5.7 and Fig. 5.8, respectively. The two figures shows that by using LLE, both of the two fitting errors are the smallest. So in the numerical test parts, LLE is chosen and applied to reduce the dimensionality.

According to the error comparison, we can draw the conclusion that, when we ap-



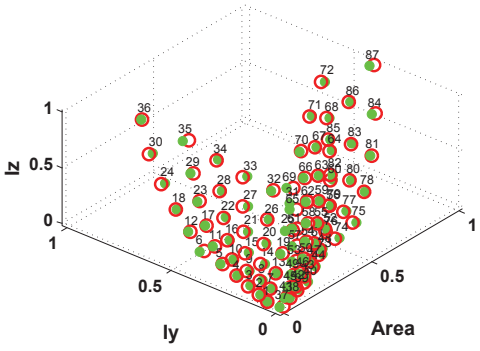
(a) PCA (b) MDS

(c) Isomap 6 (d) Isomap 10

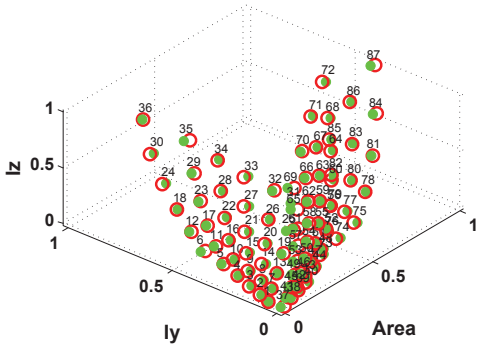
(e) Isomap 15 (f) LLE

(g) KPCA

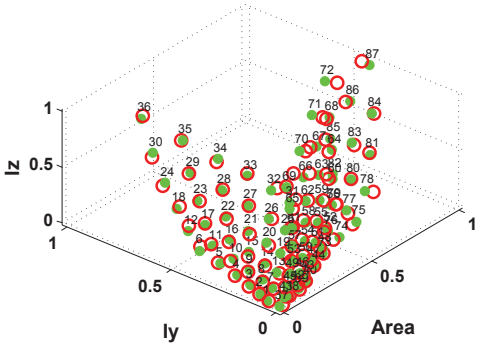
FIGURE 5.4: The reduced-order 2D design space obtained by different manifold learning methods



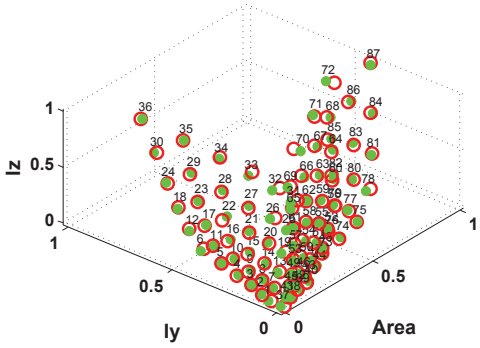
(a) PCA



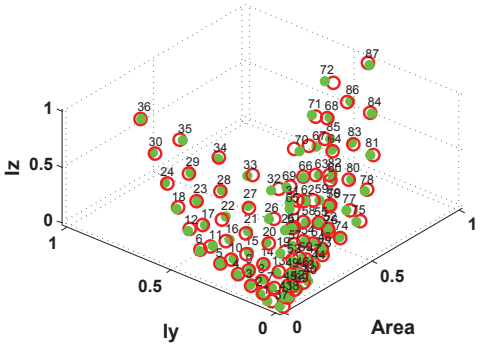
(b) MDS



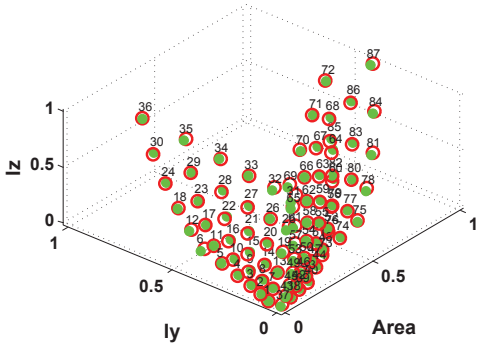
(c) Isomap 6



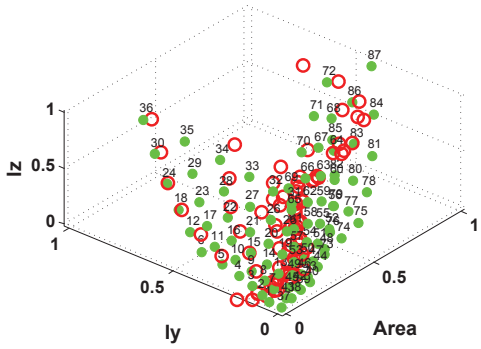
(d) Isomap 10



(e) Isomap 15

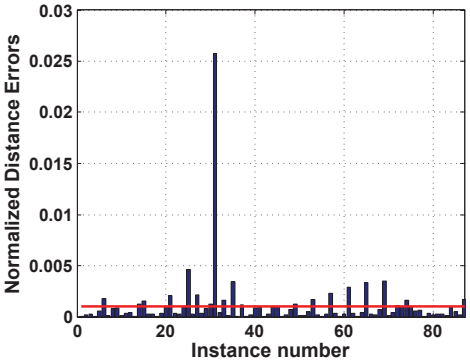


(f) LLE

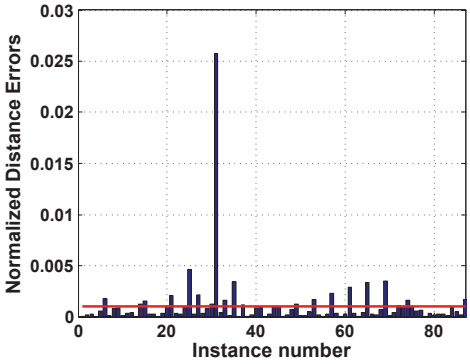


(g) KPCA

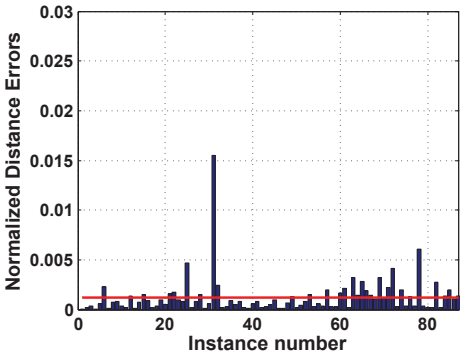
FIGURE 5.5: The comparison between the original and new 3D representation with different manifold learning methods



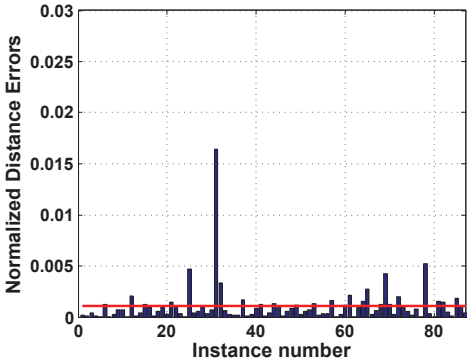
(a) PCA



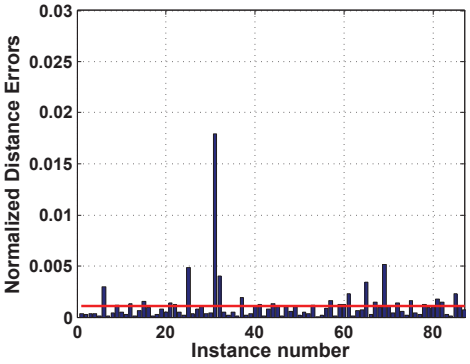
(b) MDS



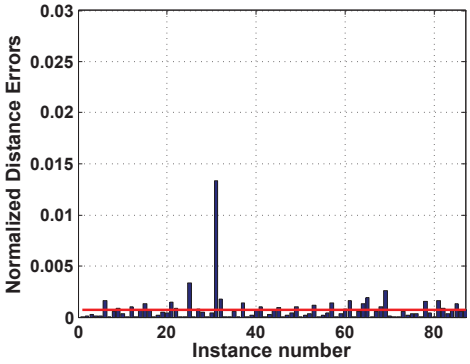
(c) Isomap 6



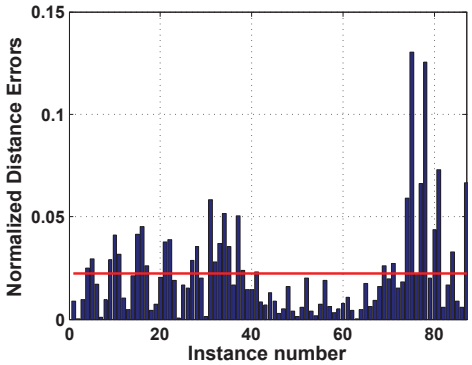
(d) Isomap 10



(e) Isomap 15



(f) LLE



(g) KPCA

FIGURE 5.6: The fitting errors for all the instances obtained by different manifold learning methods

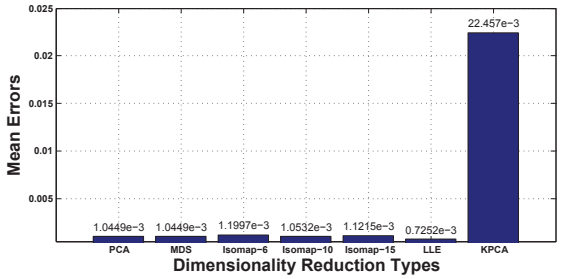


FIGURE 5.7: The mean errors of different manifold learning methods

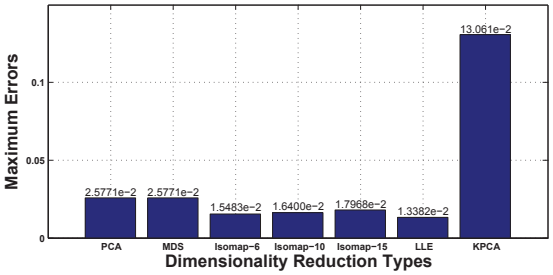


FIGURE 5.8: The maximum errors of different manifold learning methods

ply the 2nd order polynomials, both of the mean errors and the maximum errors brought by LLE are the smallest, and Isomap is the second best dimensionality reduction choice. PCA and MDS are acting the same. While KPCA brings the largest mean error and maximum error for KPCA does not preserve the relative locations of instances. As a result, we choose LLE as our default dimensionality reduction method in the numerical test section.

5.3.4 The continuous search on the manifold

After we map the manifold from the lower dimensional space R^m to the higher dimensional space R^M with polynomial fitting, any point in R^m (even if this point does not belong to the given instances) will correspond to a specific point in R^M. As a consequence, the continuous R^m will shape a continuous manifold in R^M. It is obvious that the instances in R^M will locate on the continuous manifold above. When we carry out an optimization search in the continuous space R^m, there will also be a corresponding search path on the manifold in R^M. We write the relationship of the two search paths as:

$$\mathscr{P}_m \Longleftrightarrow \mathscr{P}_M \tag{5.15}$$

where $\mathscr{P}_m$ is the optimization path in R^m, $\mathscr{P}_M$ denotes the optimization path on the manifold in R^M.
After the explicit relationship between the two kinds of design space has been built up, we carry out a gradient-based search on the manifold by using MMA. Compared with directly using MMA in the higher dimensional design space, the dimensionality reduction can help improve the optimization efficiency by reducing the size of

the search space.
We need to note that the iterative designs during the optimization process with MMA are inadmissible because we change the discrete nature of categorical optimization problem to a continuous one. But MMA always makes objective descending with designs on the continuous manifold. And due to effort of gradient information, the optimizer will find the minima with high efficiency.

5.4 Neighbour search stage

In the previous section, we have introduced that by making the design manifold continuous, the gradient-based MMA is executed to find the continuous solution. But this continuous solution is generally inadmissible since the available designs are discrete in nature, thus a rounding-off procedure is necessary to bring the inadmissible design to a admissible design, which will be elaborated in this section. We also introduce the second stage of search: the sequential neighbour search. The neighbour search procedure contains two steps: the first step is to round off the inadmissible design obtained by continuous search to an admissible design; the second step is to traverse the neighbour designs one-by-one based on the yielded admissible design.

5.4.1 Rounding-off criteria

In this chapter, we utilize the distance minimum criteria in the normalized design space to choose the nearest available design. The rounding-off performs as follows: it calculates the distance between the inadmissible design and each available instance, and selects the nearest one as the rounded-off design. This rounding-off criteria holds the advantage of being simple and low cost of calculation, however, the rounded-off admissible design may become infeasible taking into consideration the constraint functions. Another way to execute the rounding-of process is to take advantage of gradient information. Firstly, the gradients of the constrains are deduced. Then the distances between the inadmissible design with nearby available designs are calculated. Those distances are attached with weights determined by the the gradients of constraints. If the the gradients of constraints show that the nearby design is likely to violate the constraints, then the corresponding distance will be attached with a larger weight so that it becomes further from the inadmissible design. If the gradients of constraints indicate the nearby design is probably feasible, then the corresponding distance will be attached with a smaller weight, resulting in a closer neighbour distance and more chance to be chosen as the rounding-off design.

5.4.2 Neighbour search

The necessity of the neighbour search lies in the two facts: the first one is that the previous continuous search usually yields inadmissible designs; the second one is that any rounding-off process can change the performance of current design. It leads to the risk that a feasible continuous design may become infeasible, or a good continuous design may become bad.
In order to solve this problem, we propose the sequential neighbour search. Firstly, we use one single categorical variable as an example. As illustrated in Fig. 5.9, the steps of neighbour search can be listed as follows:
1) Assign neighbours to each design (instance). There are two means to determine

neighbours for a given design $\mathbf{x}^j$: R-means and k-means. The R-mean is that any design whose distance with respect to $\mathbf{x}^j$ is smaller than a predefined distance threshold R is regarded as the neighbours of $\mathbf{x}^j$, while in k-means, the k nearest adjacent designs of $\mathbf{x}^j$ are regarded as its neighbours. In our work, $\mathbf{x}^j$ is also defined as its own neighbour. Furthermore, if $\mathbf{x}^k$ is the neighbour of $\mathbf{x}^j$, then $\mathbf{x}^j$ is also forced to be the neighbour of $\mathbf{x}^k$. This measure is to preserve the symmetry of the adjacent relationship matrix. Note that this rule probably results in the inequality of neighbours numbers for different designs. Finally a bi-connecting map can be constructed.
Here if $\mathbf{x}^i$ and $\mathbf{x}^j$ are neighbours, we record this relationship as:

$$\begin{aligned} \mathbf{x}^i &\in \mathrm{Nei}(\mathbf{x}^j), \\ \mathbf{x}^j &\in \mathrm{Nei}(\mathbf{x}^i). \end{aligned} \tag{5.16}$$

It is certain that

$$\mathbf{x}^i \in \mathrm{Nei}(\mathbf{x}^j) \Longleftrightarrow \mathbf{x}^j \in \mathrm{Nei}(\mathbf{x}^i). \tag{5.17}$$

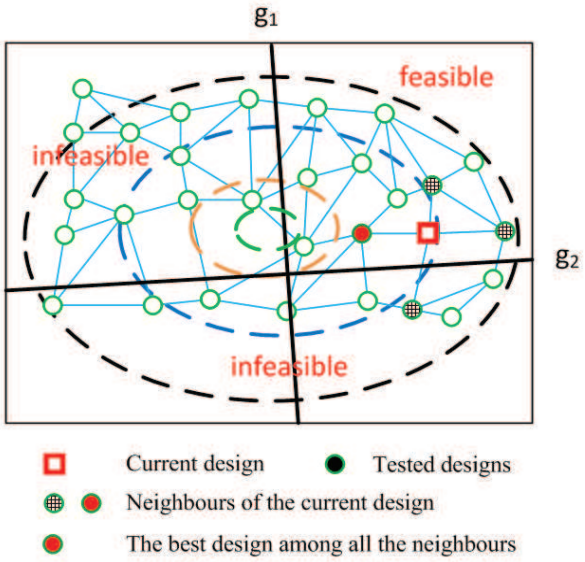
2) Evaluate neighbours of current design and select the best one as the design of next iteration. Fig. 5.9(a) shows the contour of 2-D convex objective function with the connecting map of all the instances. There are also two constraints g_1 and g_2 which limit that only several instances are feasible. We start the neighbour search from the current design, and we continue until all the neighbours are evaluated. We select the feasible neighbour with the smallest objective value as the candidate design of next iteration. If all the neighbours are infeasible, we use the weighted coefficient method to penalize the objective function with constraint functions, as shown in the following equation

$$J^* = J + k_1 \cdot \max(g_1, 0) + k_2 \cdot \max(g_2, 0) + \cdots + k_t \cdot \max(g_t, 0) \tag{5.18}$$

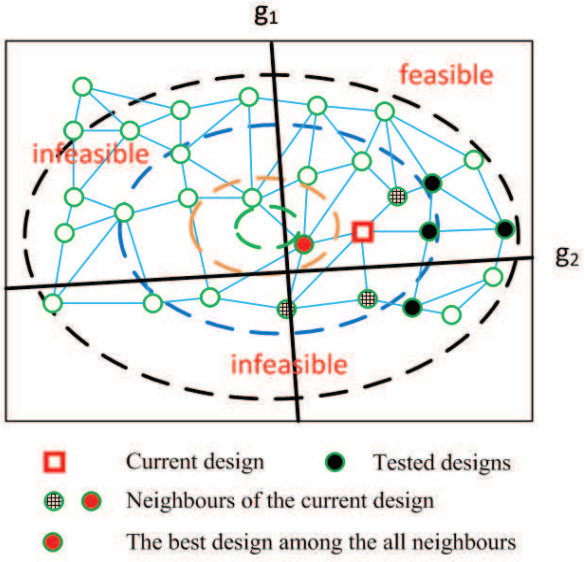
where the coefficient k_i is a large positive number. Its value relies on the relative magnitude of the constraint and objective functions, we say that k_i must be large enough that it can apparently penalize the objective when $g_i > 0$. t is the number of constraints.
Then we operate the same steps in the following iterations (Fig. 5.9(b)).
3) The iteration is stopped when no feasible better design can be found, as shown in Fig. 5.9(c).
As a categorical optimization problem may involve multiple design variables, the previous neighbour search algorithm needs to be extended. Here we assume that there are e categorical variables, and we give the definition of r-order neighbour of a design in iteration c: ${}_c\mathbf{X} = ({}_c\mathbf{x}_1, {}_c\mathbf{x}_2, \cdots, {}_c\mathbf{x}_e)$:
Definition 1: For a current design ${}_c\mathbf{X}$ with e categorical variables, if r variables among those e variables are mutated to their neighbours, respectively, ${}_c\mathbf{X} \Longrightarrow {}_w\mathbf{X}$, we call ${}_w\mathbf{X}$ is the r-order neighbour of design ${}_c\mathbf{X}$, and we record it as:

$${}_w\mathbf{X} \in \mathrm{Nei}_r({}_c\mathbf{X}) \tag{5.19}$$

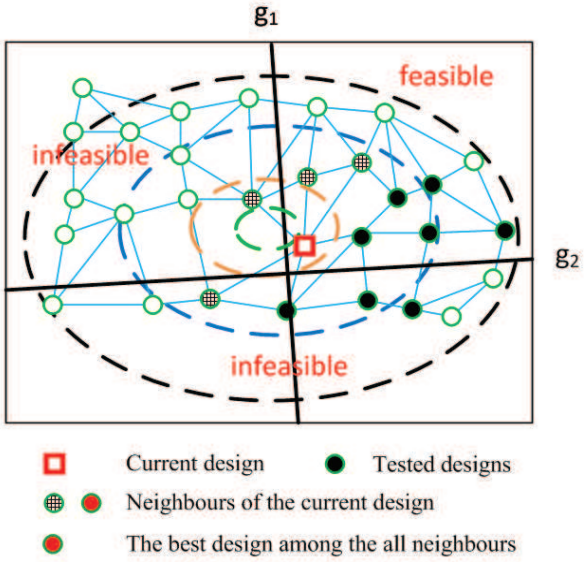
The standard steps to find all the neighbours of ${}_c\mathbf{X}$ are:
1) List all the combinations of choosing r variables among e variables. There should be C_e^r combinations. Each combination refers to a group of neighbour designs.
2) For each group, evaluate all the neighbours.
3) Select the best neighbour design as the starting design of next iteration.



(a) 1st iteration



(b) 2nd iteration



(c) 3rd iteration

FIGURE 5.9: Neighbour search for a single variable with two attributes

4) Recycle the loop 1), 2) and 3) until no better design can be found.
Definition 2: for a design ${}_c\mathbf{X}$, if all its r-order neighbours perform not better than ${}_c\mathbf{X}$, then ${}_c\mathbf{X}$ is the r-order local solution of the optimization problem.
In our neighbour search process with multiple design variables, the final solution can be proved to be a r-order local solution because we traverse all the r-order neighbours.

5.5 Numerical tests

In this section, two numerical examples, including the ten-bar truss and the 120-bar dome structure optimization problem, are experimented. The second example is also used to discuss the influence of optimization parameters and the performance of the proposed strategy as compared to other discrete algorithms.

5.5.1 The ten-bar structure optimization problem

As shown in Fig. 5.10, the 2D cantilever structure contains ten bars which are divided into four groups: the horizontal group, the vertical group, the sub-diagonal group and the principal diagonal group. The bars in the same group share the same cross-section. The left end of the structure is fixed to rigid wall while the lower right end bears a downward 10000N force. The material properties for all the bars are: Young's modulus E $= 2.1e11$Pa and the density $\rho = 7850\text{kg}/m^3$. The available cross-section types are shown in Fig. 5.2. We aim to minimize the overall strain energy under external force $\mathbf{P}$, while the mass of the whole structure should not exceed 400kg and no local buckling would happen. Note that critical forces for linear buckling can be expressed as follows:

$$f_y^{\text{cr}} = -\frac{\pi^2 \text{EI}_y}{\text{L}^2}, \quad f_z^{\text{cr}} = -\frac{\pi^2 \text{EI}_z}{\text{L}^2}. \tag{5.20}$$

As the critical buckling loads are always compression forces, we mark them with negative signs.
The optimization problem is formulated as:

$$\begin{aligned}
\text{min.:} \quad & Se(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4) = 0.5\mathbf{u}^{\text{T}}\mathbf{K}\mathbf{u}; \\
\text{s. t.:} \quad & \text{mass} - 400 \leq 0; \\
& \mathbf{K}\mathbf{u} = \mathbf{P}; \\
& \max(\mathbf{F}_y^{\text{cr}} - \mathbf{F}) \leq 0, \ \ \mathbf{F}_y^{\text{cr}} = (f_y^{\text{cr1}}, f_y^{\text{cr2}}, \cdots, f_y^{\text{cr87}}); \\
& \max(\mathbf{F}_z^{\text{cr}} - \mathbf{F}) \leq 0, \ \ \mathbf{F}_z^{\text{cr}} = (f_z^{\text{cr1}}, f_z^{\text{cr2}}, \cdots, f_z^{\text{cr87}}); \\
& \mathbf{x}_{\text{i}} \in \{\mathbf{x}_{\text{i}}^1, \mathbf{x}_{\text{i}}^2, \cdots, \mathbf{x}_{\text{i}}^{87}\}, \ \ \text{i} = 1, 2, 3, 4; \\
& \mathbf{x}_{\text{i}}^{\text{j}} = (\text{A}_{\text{i}}^{\text{j}}, \text{I}_{y\text{i}}^{\text{j}}, \text{I}_{z\text{i}}^{\text{j}})^{\text{T}}, \ \ \text{j} = 1, 2, \cdots, 87.
\end{aligned} \tag{5.21}$$

where $\mathbf{F}$ denotes the row vector composed by the inner forces of all the bars.
The initial choice for the four groups of bars is the number 15 cross-section. The number of neighbours and the neighbour order are set as 5 and 2, respectively. Fig. 5.11(a-c) shows the design histories of the objective, of the mass constraint and of the two buckling constraints. In Fig. 5.11(a-c), the design histories with solid circle markers denote the continuous search process, while the following design histories

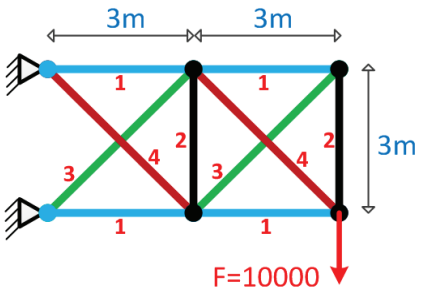
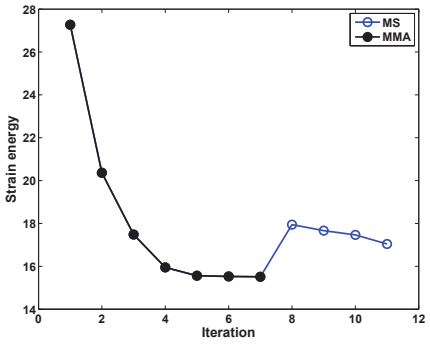


FIGURE 5.10: The ten-bar structure

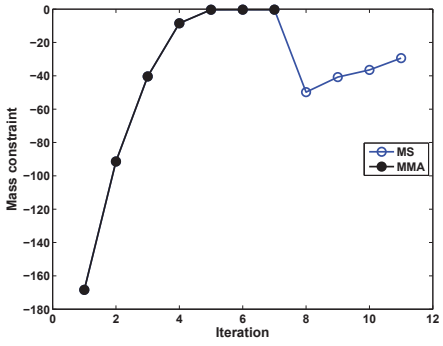
with hollow circle markers mean the neighbour search process. The final objective value is 17.0417J, with the mass constraint and buckling constraints are all satisfied but inactive, indicating that the final solution is a 2-order local solution located inside the feasible region. We can find that during the shifting from gradient-based MMA to the discrete neighbour search process, the rounding-off step causes an apparent rise of the objective and a sudden drop of the mass constraint. That can be regarded as the result of rounding-off to admissible designs.
The final design is (36 36 36 36), while their evolution design histories are displayed in the reduced-order 2D space Fig. 5.12(a-d) and their corresponding original 3D space Fig. 5.13(a-d), respectively. Similar to the history figures of the response functions, the optimization paths with solid circle markers indicate the continuous search stage, while the optimization paths with hollow circle markers denote the neighbour search process. It can be seen that the search paths in the 2D design space actually correspond to a search trajectory on the manifold in the original 3D design space. It proves the success of the dimensionality reduction strategy of the design space in handling categorical optimization problem, with the application of gradient-based MMA.

Then we restart the optimization with a different initial design (14 14 14 14). The design histories of the objective, the mass constraint and the buckling constraints are displayed in Fig. 5.14(a-c), respectively. We could find that finally all the design constraints are satisfied. Among them, the mass constraint is nearly active. The final design is (78 46 48 77) with the strain energy value 14.2483J which is smaller than that in the previous test. The corresponding optimization paths for all the variables in 2D space and 3D space are shown in Fig. 5.15(a-d) and Fig. 5.16(a-d), respectively. This test also proves that even for the simple ten-bar structure design problem, at least two 2-order local solutions exist.

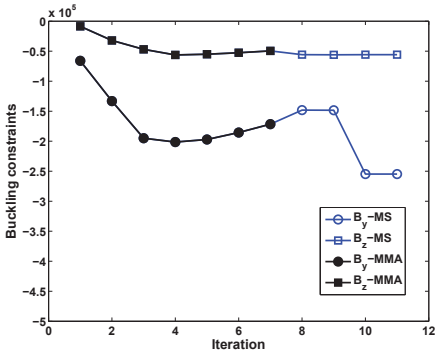
In the third test, we give a random initial design (12 13 50 53). After the continuous search and neighbour search processes, the final solution is (36 70 48 78), and the final objective value is 16.3481J. All the constraints are satisfied while the mass constraint is roughly active, as the same as in the previous test. We also show the design paths in the two kinds of design space as in Fig. 5.18(a-d) and Fig. 5.19(a-d). In this test, we obtain the third 2-order local solutions, indicating that the categorical optimization problems brings extra complexity compared with other kinds of optimization problems, even when the objective is convex and the active constraint is linear.



(a) The Objective

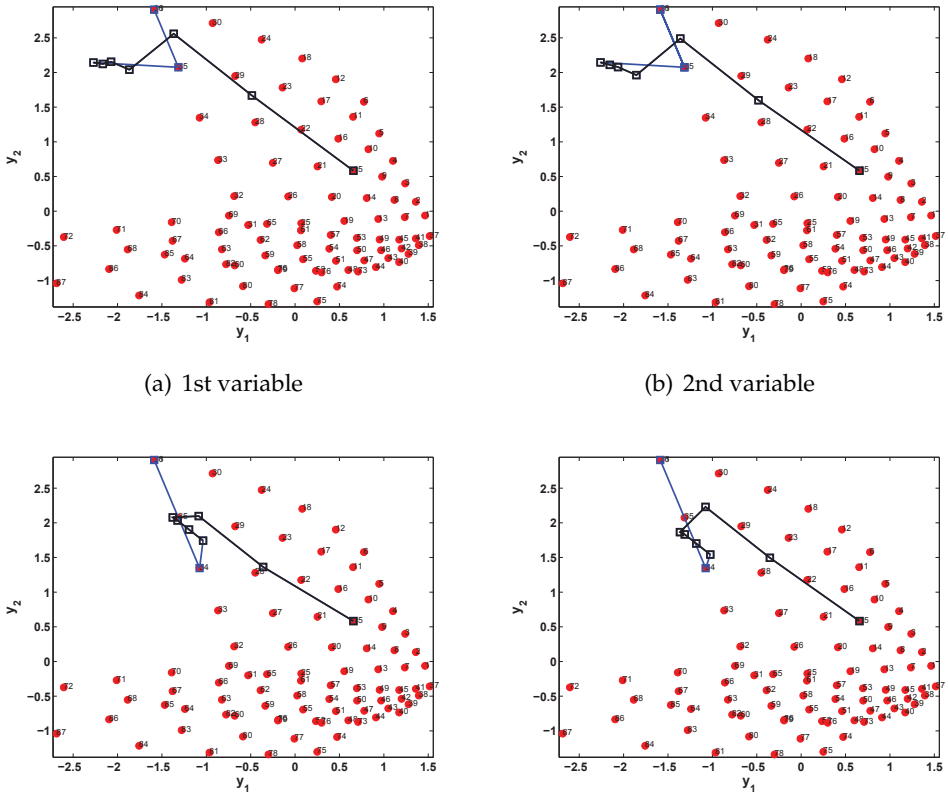


(b) The mass constraint



(c) The two buckling constraints

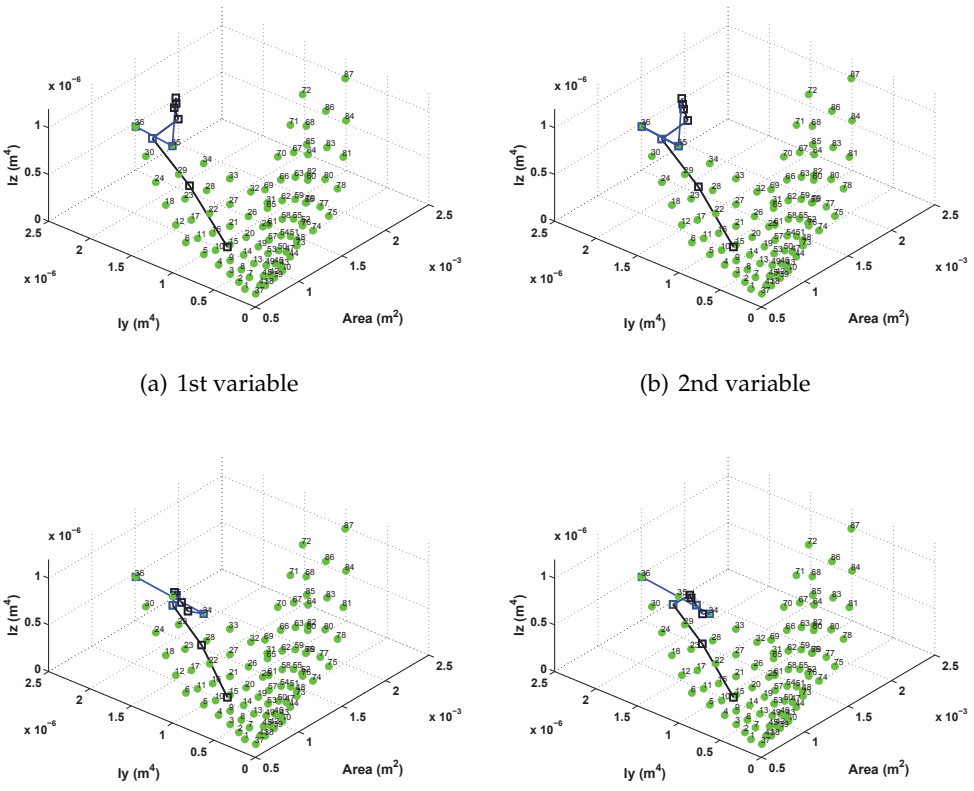
FIGURE 5.11: The design history of the objective and constraints



(a) 1st variable (b) 2nd variable

(c) 3rd variable (d) 4th variable

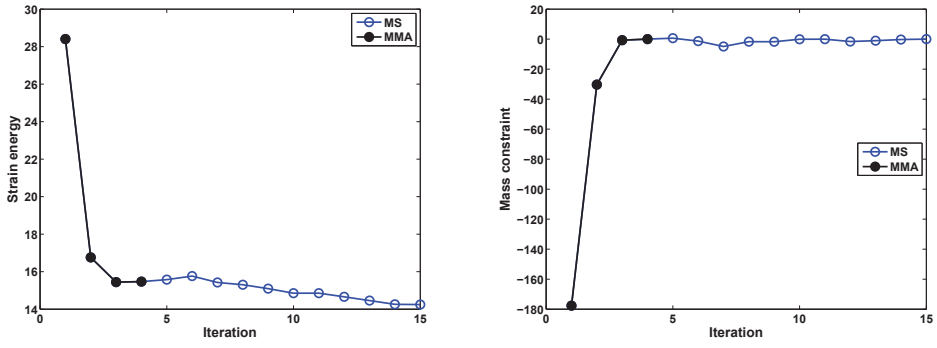
FIGURE 5.12: The evolution history of four variables in 2D space



(a) 1st variable (b) 2nd variable

(c) 3rd variable (d) 4th variable

FIGURE 5.13: The evolution history of four variables in the original 3D space

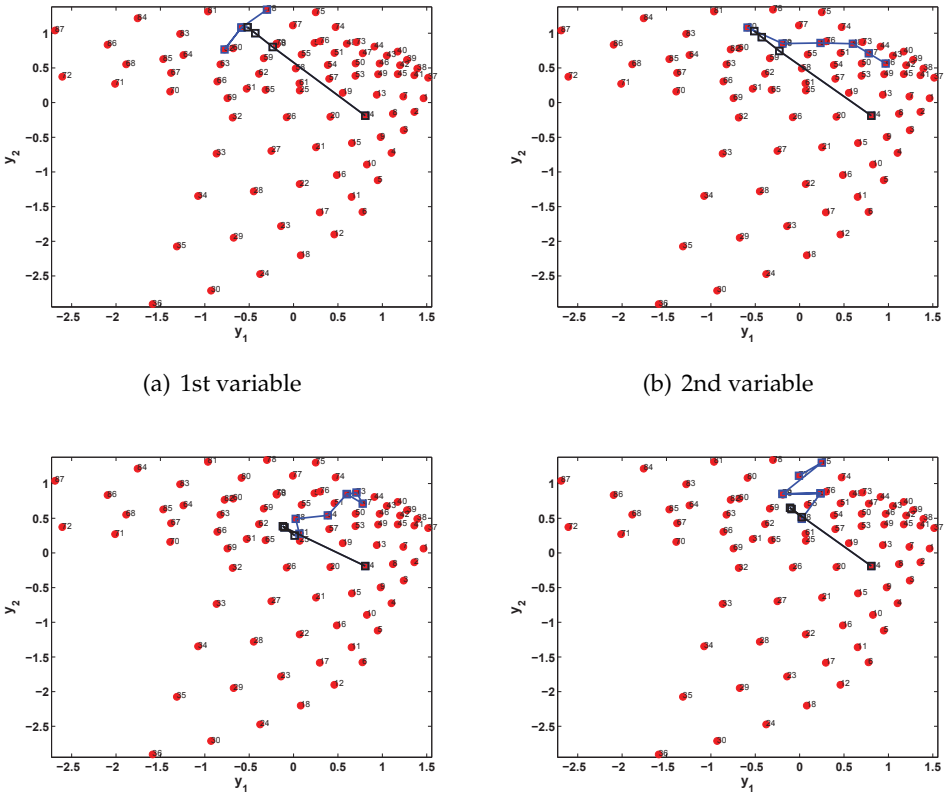


(a) The Objective

(b) The mass constraint

(c) The two buckling constraints

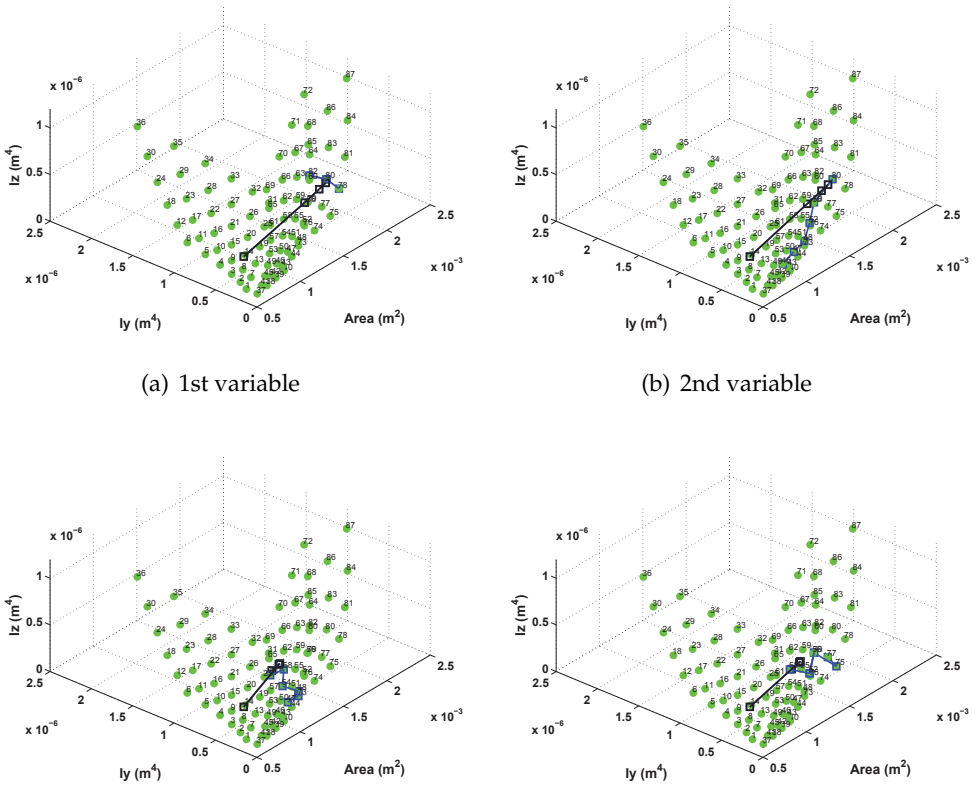
FIGURE 5.14: The design history of the objective and constaints



(a) 1st variable (b) 2nd variable

(c) 3rd variable (d) 4th variable

FIGURE 5.15: The evolution history of four variables in 2D space



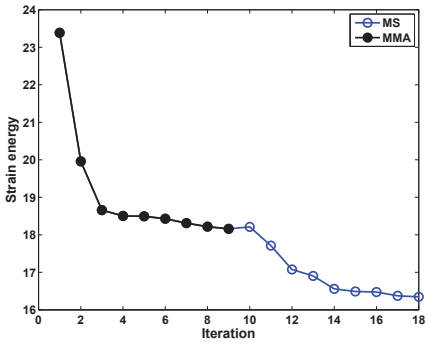
(a) 1st variable

(b) 2nd variable

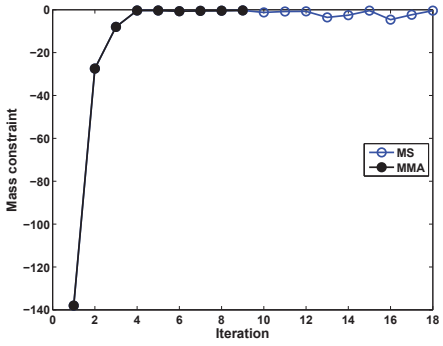
(c) 3rd variable

(d) 4th variable

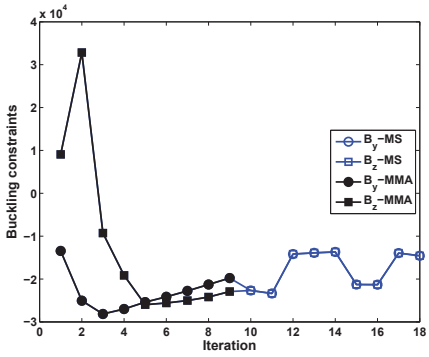
FIGURE 5.16: The evolution history of four variables in the original 3D space



(a) The Objective

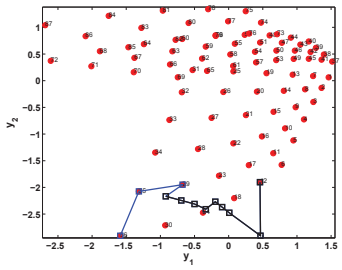


(b) The mass constraint

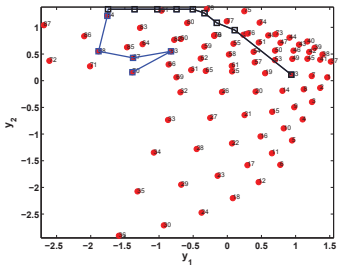


(c) The two buckling constraints

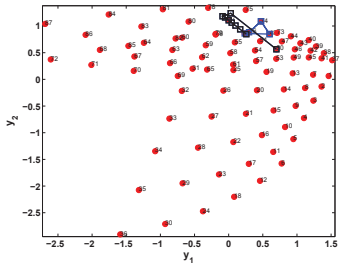
FIGURE 5.17: The ten-bar truss design: the design history of the objective and constraints



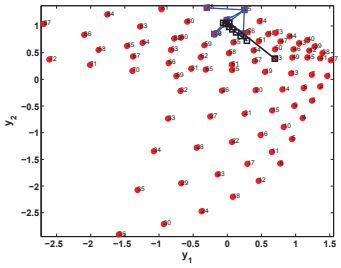
(a) 1st variable



(b) 2nd variable

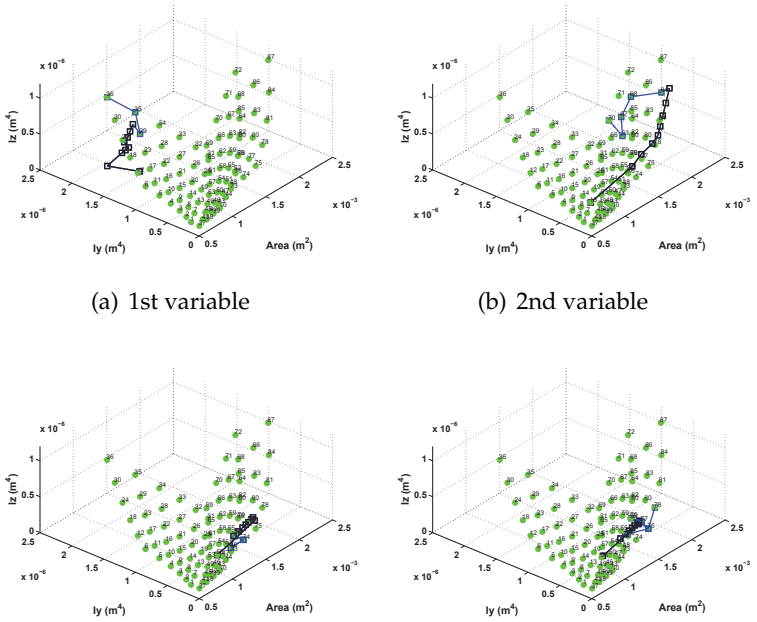


(c) 3rd variable



(d) 4th variable

FIGURE 5.18: The ten-bar truss design: the evolution history of four variables in 2D space



(a) 1st variable (b) 2nd variable

(c) 3rd variable (d) 4th variable

FIGURE 5.19: The ten-bar truss design: the evolution history of four variables in the original 3D space

5.5.2 The dome structure design problem

In the second test, we optimize a dome structure (Fig. 5.20) which consists of 120 bars and is under vertical loads. The base of the dome is fixed on the ground. The bars are divided into seven groups, and each group is described with one categorical variable. The material remains the same as in the first example. The objective is to minimize the strain energy, while the mass constraint and the buckling constraints are obeyed. The available bar cross-sections are as the same as those in the previous test. The optimization problem is stated as:

$$
\begin{aligned}
\text{min.:}\quad & Se(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_7) = 0.5\mathbf{u}^{\mathrm{T}}\mathbf{K}\mathbf{u}; \\
\text{s. t.:}\quad & \text{mass} - 6000 \leq 0; \\
& \mathbf{K}\mathbf{u} = \mathbf{P}; \\
& \max(\mathbf{F}_y^{\mathrm{cr}} - \mathbf{F}) \leq 0, \;\; \mathbf{F}_y^{\mathrm{cr}} = (f_y^{\mathrm{cr1}}, f_y^{\mathrm{cr2}}, \cdots, f_y^{\mathrm{cr120}}); \\
& \max(\mathbf{F}_z^{\mathrm{cr}} - \mathbf{F}) \leq 0, \;\; \mathbf{F}_z^{\mathrm{cr}} = (f_z^{\mathrm{cr1}}, f_z^{\mathrm{cr2}}, \cdots, f_z^{\mathrm{cr120}}); \\
& \mathbf{x}_i \in \{\mathbf{x}_i^1, \mathbf{x}_i^2, \cdots, \mathbf{x}_i^{87}\}, \;\; i = 1, 2, \cdots, 7; \\
& \mathbf{x}_i^j = (A_i^j, I_{yi}^j, I_{zi}^j)^{\mathrm{T}}, \;\; j = 1, 2, \cdots, 87.
\end{aligned}
\tag{5.22}
$$

For the optimization, the neighbour number and the search order number are set to be 5 and 2, respectively. The initial design is (13 13 13 13 13 13 13). The final objective value is 2.9934J, as shown in Fig. 5.21(a), while both of the mass constraint and the buckling constraints are satisfied. Among them, the mass constraint is active in the final design, as shown in Fig. 5.21(b,c). The final design is (58 86 55 53 37 77 59), and the optimization paths in two kinds of design space are illustrated in Fig. 5.22 and Fig. 5.23, respectively.

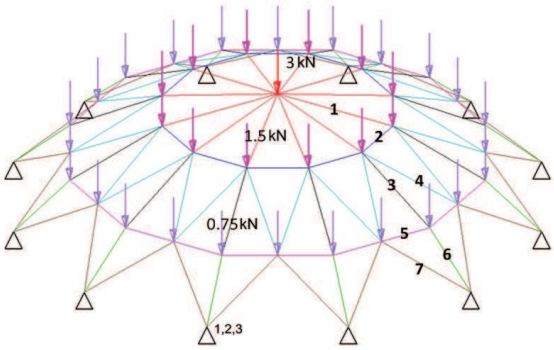


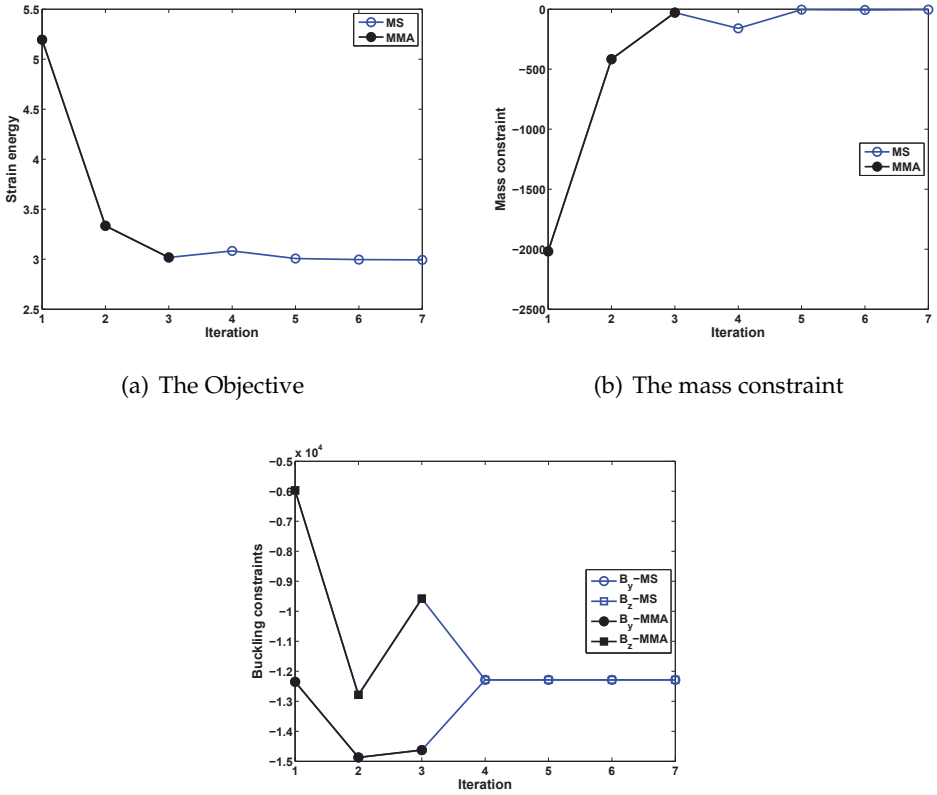
FIGURE 5.20: The dome structure

In this test, we can also find even though the mass constraint and the buckling constraints are all satisfied, only the mass constraint is active.

5.5.3 Effects of neighbour search parameters

In order to discuss how the discrete search parameters, namely the numbers of neighbours N_{nei} and the number of simultaneously changing variables k (namely the search order), influence the optimization results, a series of tests are executed with different combinations of the two search parameters. We set the number of neighbours as 4, 5, 6 and 7 respectively, and the values of search orders are 1, 2, 3, 4 and 5. Tab. 5.2 shows the final objective values obtained by different combinations of search parameters. It can be found that the increasing values of search orders can help decrease the objective values at the beginning, generally from 1 to 3, but when we continue increasing the value of search order, the final solutions goes worse. Furthermore, the increasing values of search orders will result in a significant rise of function evaluations, as shown in Tab. 5.3. If we fix the value of search orders and vary the number of search neighbours, we can also find a similar phenomenon, namely aiming at obtaining better optimization solutions, the increasing of neighbour number may work at the beginning (from 4 to 5), but fails when it keep rising (from 5 to 7). It is also obvious that the increasing number of search neighbours also cause a significant rising the evaluation numbers, as listed in Tab. 5.3. The corresponding final feasible solutions obtained by each combination are shown in Tab. 5.4.

It can be seen that the blindly increase of either the mutated variable number or the search neighbour number, will lead to some kinds of uncertainties in the optimization results: It may go in vain aiming at obtaining better designs even if paying for the price of large time cost for the function evaluations. The reasons of this phenomenon lie in the mechanism of mutation search and the optimization problem itself. As the final design of each combination must be a k-order local solution, we can draw the conclusion that the optimization problem contains multiple k-order local solutions. And this is the first reason why increasing the number of mutated variables or search neighbours may obtain worse result. The second reason is that compared with the optimization path formed by smaller numbers of mutated variables or search neighbours, the optimization path with larger numbers of mutated variables or search neighbours will change to a different one, leading to a totally different k-order local solution which may perform worse. From another side, by

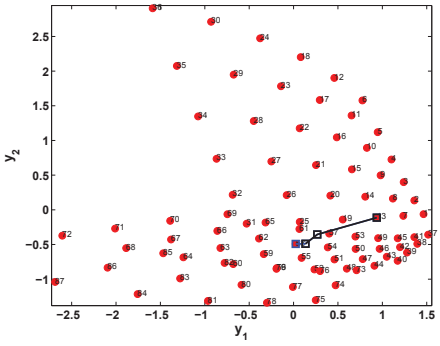


(a) The Objective

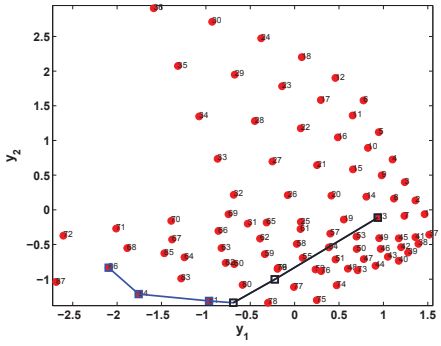
(b) The mass constraint

(c) The two buckling constraints

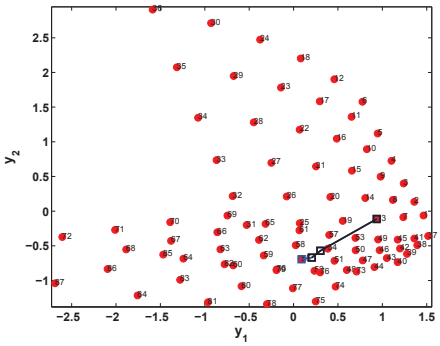
FIGURE 5.21: The dome structure design: the design history of the objective and constraints



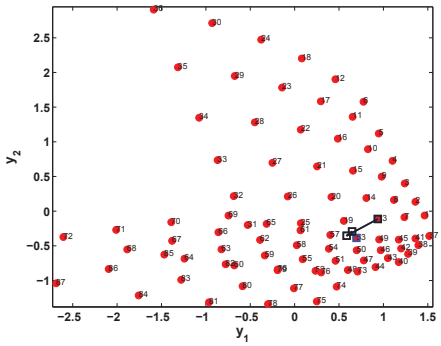
(a) 1st variable



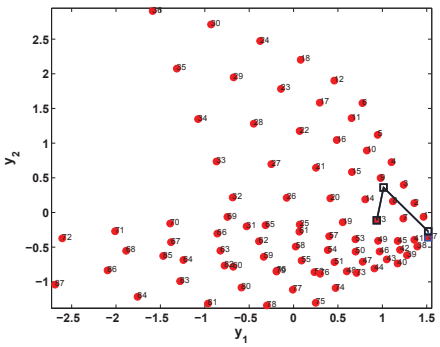
(b) 2nd variable



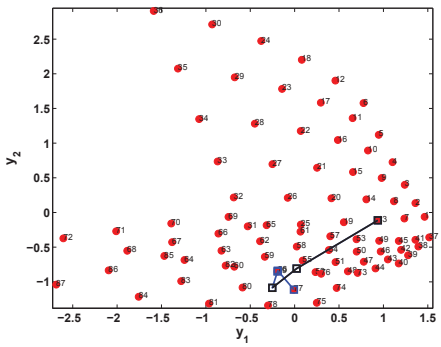
(c) 3rd variable



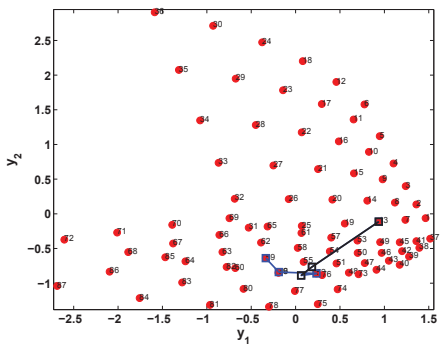
(d) 4th variable



(e) 5th variable

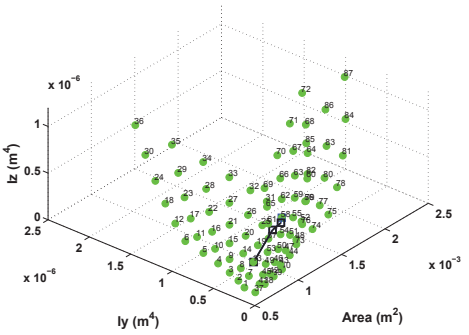


(f) 6th variable

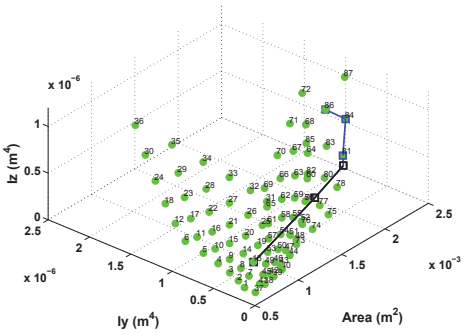


(g) 7th variable

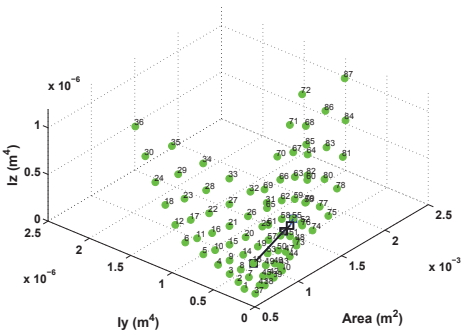
FIGURE 5.22: The dome structure design: the evolution history of four variables in 2D space



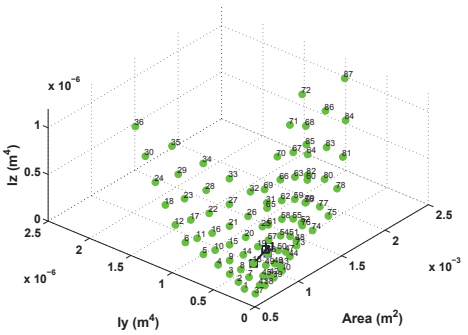
(a) 1st variable



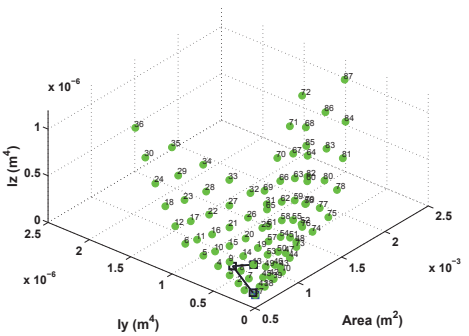
(b) 2nd variable



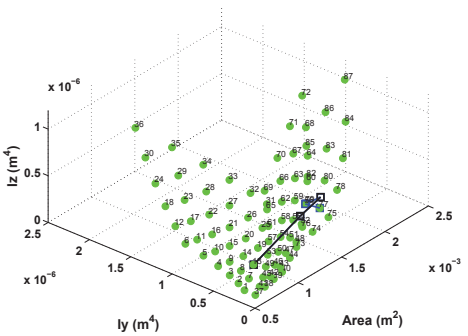
(c) 3rd variable



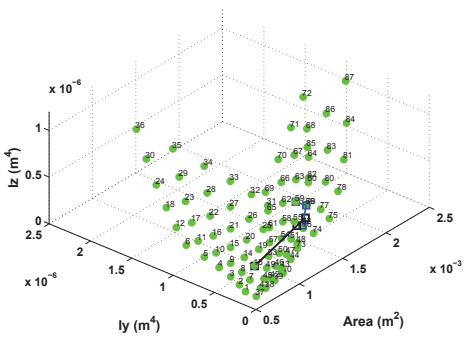
(d) 4th variable



(e) 5th variable



(f) 6th variable



(g) 7th variable

FIGURE 5.23: The dome structure design: the evolution history of four variables in the original 3D space

using small numbers of the mutated variable or the search neighbours and carrying on several optimization with different initial designs, may be a good strategy to deal with the problem.

TABLE 5.2: The influence of search parameters

	$N_{nei} = 4$	$N_{nei} = 5$	$N_{nei} = 6$	$N_{nei} = 7$
$k = 1$	3.0330	3.0291	3.0574	3.0574
$k = 2$	2.9954	2.9873	2.9874	2.9897
$k = 3$	2.9860	2.9879	2.9856	2.9863
$k = 4$	2.9995	2.9883	2.9869	2.9859
$k = 5$	2.9976	2.9871	2.9892	2.9860

TABLE 5.3: The corresponding numbers of function evaluations

	$N_{nei} = 4$	$N_{nei} = 5$	$N_{nei} = 6$	$N_{nei} = 7$
$k = 1$	56	72	66	138
$k = 2$	1318	2652	2737	3182
$k = 3$	11900	9143	35219	34613
$k = 4$	14608	60730	195467	432655
$k = 5$	46527	190634	652063	1390035

TABLE 5.4: Final feasible designs

	$N_{nei} = 4$	$N_{nei} = 5$	$N_{nei} = 6$	$N_{nei} = 7$
$k = 1$	(58,81,79,19, 1,79,52)	(58,81,55,19, 45,60,52)	(58,81,55,19, 49,79,52)	(58,81,55,19, 49,79,52)
$k = 2$	(58,86,76,53, 37,77,74)	(65,87,58,53, 37,75,74)	(33,87,58,14, 37,80,52)	(58,87,51,14, 37,75,59)

$k = 3$	(69,87,58,53, 37,60,52)	(31,84,58,19, 37,80,59)	(51,84,48,15, 37,82,59)	(58,84,51,11, 37,82,59)
$k = 4$	(8,81,58,53, 1,78,74)	(58,87,58,19, 37,82,76)	(51,87,32,19, 37,75,74)	(1,87,69,46, 37,82,52)
$k = 5$	(69,81,58,53, 37,78,74)	(58,87,48,19, 37,82,76)	(48,84,48,53, 37,75,59)	(65,87,73,14, 37,80,74)

5.5.4 Comparison of results

To compare the proposed method with other methods, we also carry out the evolution strategy with covariance matrix adaptation (CMA-ES) and the simulated annealing (SA) with the 120-bar dome structure example. The geometrical parameters of the dome structure, the materials and the load conditions remain the same. We also set the same initial design. Note that both of the two comparison algorithm are based on the random search.

We choose the result group with k=2 as our experimental set. The corresponding numbers of function evaluations are 1318, 2652, 2737 and 3148, respectively. So when applying CMA-ES and SA in solving the dome structure design problem, we also limit the upper bounds of the function evaluation numbers as 1318, 2652, 2737 and 3148. Each set will run 5 times in order to yield the mean values.

The optimization results including the final objective, the constraint functions and the final designs obtained by CMA-ES with different function evaluation numbers are listed in Tab. 5.5, 5.6, 5.7 and 5.8, and those optimization results yielded with SA are illustrated in Tab. 5.9, 5.10, 5.11 and 5.12. We can see that all the final designs are feasible, but CME-ES cannot meet the bounds of the constraints. The SA algorithm takes a discrete random search, so it seems to perform better than CMA-ES to deal with this kind of problems.

We take the average values of objective functions obtained by the two methods, and compare them with the proposed MMA-MS method, as shown in Tab. 5.13. We can see that with the same number of function evaluations, SA is better than CMA-ES, but the two-stage MMA-MS performs the best. The reason is that MMA-MS applies the continuous approximate search on the reduced order manifold at first, resulting in a big decrease of the objective value, then it utilizes a fine local search to find a local solution based on the good solution obtained with continuous search. While CMA-ES performs the worst because it does not reduce the design space size, and its final continuous brings large error when it is rounded-off to admissible solutions. SA can make a discrete search, so it always produces admissible solutions, but in the final stage, it cannot converge to those better local solutions due to its evolutionary working mechanism with a limited number of function calls.

TABLE 5.5: Optimization results by CMA-ES (1318 evaluations)

	Se(J)	MC(kg)	BCY(N)	BCZ(N)	Final design

1st	6.6989	-1853.1	-1542.8	-1542.8	(37,37,37,67,37,37,37)
2nd	5.7570	-45.8	-1542.8	-1542.8	(37,37,37,87,37,87,37)
3rd	5.7240	-45.8	-2325.0	-2325.0	(84,37,37,37,87,87,37)
4th	6.5901	-856.0	-1542.8	-1542.8	(37,37,87,37,87,37,37)
5th	4.7264	-365.1	-2324.6	-2324.6	(12,87,37,37,87,84,37)

TABLE 5.6: Optimization results by CMA-ES (2652 evaluations)

	Se(J)	MC(kg)	BCY(N)	BCZ(N)	Final design
1st	5.7219	-798.0	-1542.8	-1542.8	(37,37,37,37,6,37,87)
2nd	5.3553	-903.3	-1542.4	-1542.4	(37,36,37,37,87,87,37)
3rd	4.4868	-352.0	-2325.5	-2325.5	(87,23,56,37,37,87,37)
4th	5.0231	-519.2	-1542.5	-1542.5	(37,87,37,37,87,87,37)
5th	6.0247	-754.3	-2352.2	-2352.2	(87,37,87,37,37,37,37)

TABLE 5.7: Optimization results by CMA-ES (2737 evaluations)

	Se(J)	MC(kg)	BCY(N)	BCZ(N)	Final design
1st	5.7674	-1063.7	-2324.8	-2324.8	(87,37,37,37,37,87,37)
2nd	5.7478	-966.0	-1542.8	-1542.8	(37,37,37,37,37,37,87)
3rd	5.3031	-1231.6	-2324.6	-2324.6	(87,87,37,37,37,37,37)
4th	5.8087	-675.6	-1542.8	-1542.8	(37,37,37,83,37,87,37)
5th	4.4720	-521.9	-1542.5	-1542.5	(37,87,37,37,41,37,84)

TABLE 5.8: Optimization results by CMA-ES (3182 evaluations)

	Se(J)	MC(kg)	BCY(N)	BCZ(N)	Final design
1st	4.7445	-331.0	-2352.3	-2352.3	(86,87,87,37,37,37,37)
2nd	4.7318	-425.7	-2324.6	-2324.6	(17,87,37,37,87,81,37)
3rd	4.8546	-269.3	-1542.5	-1542.5	(37,87,77,68,41,37,37)
4th	4.9804	-1530.8	-1542.4	-1542.4	(37,55,37,37,37,37,71)
5th	5.3499	-186.7	-1542.5	-1542.5	(37,87,37,63,87,37,37)

TABLE 5.9: Optimization results by SA (1318 evaluations)

	Se(J)	MC(kg)	BCY(N)	BCZ(N)	Final design
1st	3.6430	-176.9	-43834	-165.1	(17,56,68,6,59,16,23)
2nd	3.3475	-3.4	-4679.6	-4679.6	(75,78,74,21,54,36,40)
3rd	3.8807	-812.0	-19841	-1645.3	(12,36,68,5,42,29,19)
4th	3.2532	-58.7	-7180.0	-7180.0	(73,78,47,73,49,17,74)
5th	3.6693	-50.2	-11549	-11549	(80,11,63,27,38,44,48)

TABLE 5.10: Optimization results by SA (2652 evaluations)

	Se(J)	MC(kg)	BCY(N)	BCZ(N)	Final design
1st	3.6134	-372.2	-15433	-6042.3	(17,30,52,22,21,16,35)
2nd	3.4196	-37.0	-11877	-8173.6	(52,18,28,16,25,59,56)
3rd	3.1676	-100.3	-5473	-5473.5	(25,67,64,42,37,70,56)
4th	3.1872	-64.3	-4111	-4111.2	(73,81,17,39,61,62,79)
5th	3.1996	-20.5	-9501	-9501.2	(61,64,47,47,27,66,52)

TABLE 5.11: Optimization results by SA (2737 evaluations)

	Se(J)	MC(kg)	BCY(N)	BCZ(N)	Final design
1st	3.1252	-24.4	-6470.4	-6470.4	(52,64,52,45,38,84,34)
2nd	3.0967	-10.7	-11137	-10205	(76,67,65,25,4,75,36)
3rd	3.9301	-46.9	-4211.9	-4211.9	(31,63,70,76,79,76,41)
4th	3.3107	-89.9	-16981	-16981	(69,36,33,61,38,26,77)
5th	3.6331	-37.2	-16153	-13782	(28,12,57,65,76,74,36)

TABLE 5.12: Optimization results by SA (3182 evaluations)

	Se(J)	MC(kg)	BCY(N)	BCZ(N)	Final design
1st	3.2023	-36.8	-18475	-13616	(59,64,35,19,61,56,24)
2nd	3.3779	-6.3	-11137	-11137	(76,29,79,50,27,52,69)
3rd	3.6396	-8.3	-42121	-8431.8	(33,36,24,24,80,27,17)
4th	3.3301	-5.1	-9580.7	-9580.7	(23,36,25,73,21,29,56)
5th	3.4755	-64.6	-7200.3	-7200.3	(47,81,65,53,85,24,44)

TABLE 5.13: Objective comparison obtained by three methods

	1318 runs	2652 runs	2737 runs	3182 runs
CMA-ES	5.8993	5.3224	5.4198	4.9322
SA	3.5588	3.3175	3.4192	3.4052
MMA-MS	2.99542	2.9874	2.98744	2.9897

5.6 A fast neighbour search algorithm

The neighbour search algorithm elaborated in section 5.4 can find a r-order local solution, but for large numbers of neighbours and neighbour search order, the number of simulations will be massive, resulting in excessive time cost. In this section, we propose a modified fast neighbour search algorithm based on the gradient information which can reduce the number of simulations significantly and bring appreciable time-saving benefits.
We have already defined the penalized objective as follows

$$J^* = J + k_1 \cdot \max(g_1, 0) + k_2 \cdot \max(g_2, 0) + \cdots + k_t \cdot \max(g_t, 0) \tag{5.23}$$

The coefficient k_i should be a positive number, and k_i must be large enough so that it can apparently make a difference when $g_i > 0$.
We remark

$$g_i^c = \max(g_i, 0), \quad j = 1, 2, \cdots, t. \tag{5.24}$$

Then Eq. 5.23 can be written as

$$J^* = J + \sum_{i=1}^{t} k_i \cdot g_i^c \tag{5.25}$$

Next we will derive the gradients of the modified objective J^*. For a current design $\mathbf{x}^0 = (\mathbf{x}_1^0, \mathbf{x}_2^0, \cdots, \mathbf{x}_e^0)$, we record the j-th design variable $\mathbf{x}_j^0$ with M attributes as:

$$\mathbf{x}_j^0 = ({}^1a_j^0, {}^2a_j^0, \cdots, {}^Ma_j^0)^T \tag{5.26}$$

suppose the design space is continuous, then the gradient of the original objective function J can be expressed as

$$\frac{\partial J}{\partial \mathbf{x}_j^0} = \left(\frac{\partial J}{\partial ({}^1a_j^0)}, \frac{\partial J}{\partial ({}^2a_j^0)}, \cdots, \frac{\partial J}{\partial ({}^Ma_j^0)}\right)^T. \tag{5.27}$$

The gradient of the i-th constraint function g_i can be written as

$$\frac{\partial g_i}{\partial \mathbf{x}_j^0} = \left(\frac{\partial g_i}{\partial ({}^1a_j^0)}, \frac{\partial g_i}{\partial ({}^2a_j^0)}, \cdots, \frac{\partial g_i}{\partial ({}^Ma_j^0)}\right)^T. \tag{5.28}$$

Noting that constraint function g_i has singularity when it equals 0. In this case, we define its corresponding derivatives are 0 to fulfil the sensitivity calculation.
The gradient of the modified constraint function g_i^c can be written as

$$\frac{\partial g_c^i}{\partial \mathbf{x}_j^0} = \begin{cases} \frac{\partial g_i}{\partial \mathbf{x}_j^0} \frac{g_i^c}{g_i}, & g_i \neq 0, \\ ({}^10, {}^20, \cdots, {}^M0)^T, & g_i = 0. \end{cases} \tag{5.29}$$

Finally, combining Eq. 5.27 and Eq. 5.29, the gradient of the penalized objective

function J^* can be written as

$$\frac{\partial J^*}{\partial x_j^0} = \frac{\partial J}{\partial x_j^0} + \sum_{i=1}^{t} k_i \cdot \frac{\partial g_i^c}{\partial x_j^0}. \tag{5.30}$$

Suppose the instance x_j^t is a neighbour of x_0^j, namely

$$x_j^t \in \mathrm{Nei}(x_j^0), \tag{5.31}$$

ans we make the assumption that x_j^t is close enough to x_j^0 so that the direction of gradient $\frac{\partial J^*}{\partial x_j^0}$ remains unchanged in the small local continuous design region, thus we define

$$\begin{aligned} \Delta x_j &= x_j^t - x_j^0, \\ s(x_j^t, x_j^0) &= (\Delta x_j)^T \cdot \frac{\partial J^*}{\partial x_j^0}. \end{aligned} \tag{5.32}$$

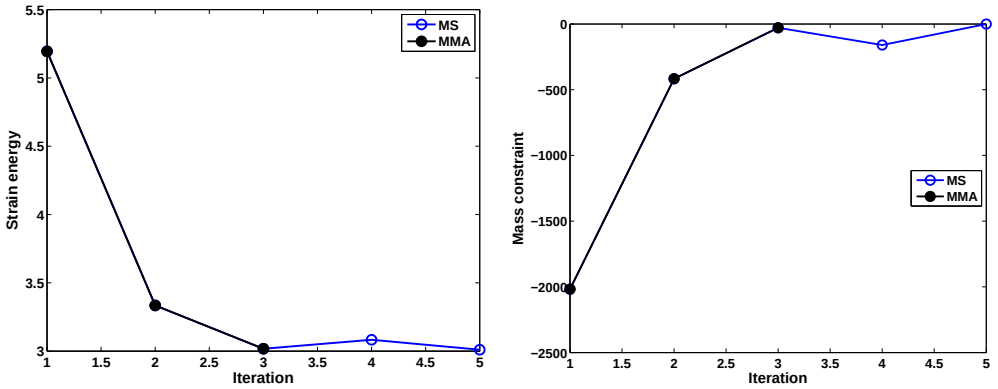
Based on the previous assumptions, we can summarize that, if $s(x_j^t, x_j^0)$ is negative, it means that changing x_j^0 to x_j^t will make the modified objective J^* descending; if $s(x_j^t, x_j^0)$ is positive, the change from x_j^0 to x_j^t will result in the increase of J^*; if $s(x_j^t, x_j^0)$ equals 0, it indicates that replacing x_j^0 by x_j^t will not affect the value of J^*. We need to mention that, since x_j^0 is also regarded as its own neighbour, in this case, $s(x_j^t, x_j^0)$ will always equal 0.
Based on the value of $s(x_j^t, x_j^0)$, the neighbours of x_j^0 are divided into two sets. We define the effective set of neighbours N_j^1 for x_j^0:

$$N_j^1 = \{x_j^t | x_j^t \in \mathrm{Nei}(x_j^0), s(x_j^t, x_j^0) \leq 0\}, \quad j = 1, 2, \cdots, e. \tag{5.33}$$

And the ineffective set is defined as:

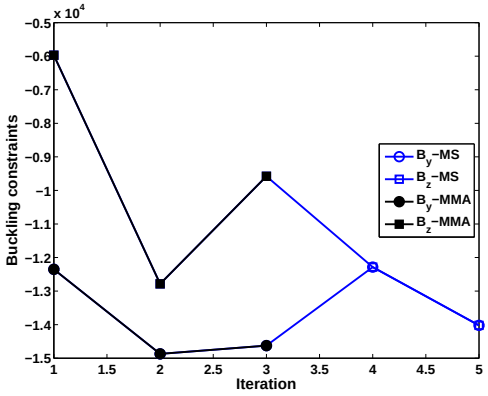
$$N_j^2 = \{x_j^t | x_j^t \in \mathrm{Nei}(x_j^0), s(x_j^t, x_j^0) > 0\}, \quad j = 1, 2, \cdots, e. \tag{5.34}$$

We need to note that, the effective set N_j^1 is never empty since it always contains the element x_j^0.
The fast neighbour search algorithm takes the procedures of the neighbour search illustrated in Section 5.4.1. The difference between them lies in that the fast neighbour search algorithm only evaluates the neighbours which belong to the effective sets, and those which belong to ineffective sets are abandoned. Consequently, the size of neighbour combinations for multiple variables will be significantly reduced, resulting in a large time saving, and the new designs can always insure the decreasing of the modified objective J^*.
We utilize the dome structure design problem in Section 5.5.2 as a testing example. The load case optimization parameters, including the number of neighbours and the searching orders, are as the same as those in Section 5.5.2. In the initialization, all the bars are set to the cross-section with number 13. The design histories of the objective, the mass constraint and two buckling constraints are displayed in Fig. 5.24(a-c), respectively. We can find that the fast neighbour search algorithm is able to find a admissible and feasible solution within a few iterations, showing its optimization



(a) The Objective

(b) The mass constraint



(c) The two buckling constraints

FIGURE 5.24: The design history of the objective and constraints

efficiency. The corresponding design paths for seven variables both in the reduced order space and original design space are illustrated in Fig. 5.25(a-g) and Fig. 5.26(a-g), proving that the fast neighbour search is a valid method.

In order to analyse the search efficiency of the fast neighbour search, we compare it with the previous neighbour search approach, by changing the number of neighbours N_{nei} and the neighbour search orders k. Firstly, the number of neighbours is set as 5, and the neighbour search orders vary from 1 to 5. The optimization objectives obtained by the original neighbour search (NS) and the fast neighbour search (FNS) are shown in Tab. 5.14, and the numbers of needed simulation are listed in Tab. 5.15. We can find that the objective values obtained by FNS are just slightly larger than those obtained by NS, but the evaluation numbers with FNS are much less than those of NS, indicating FNS has a much higher optimization efficiency and is especially suitable for optimization problems with large scale of variables.
We also fix the number of search orders and change N_{nei} from 4 to 7. The final objective values and evaluation numbers are shown in Tab. 5.16 and 5.17, respectively. We can also draw the conclusion that FNS brings considerable reduction in time cost with just a slight increase of objective values.

TABLE 5.14: Optimization objectives by neighbour search (NS) and the fast neighbour search (FNS)

k	1	2	3	4	5
NS	3.0291	2.9873	2.9879	2.9883	2.9871
FNS	3.0027	3.0035	3.0035	3.0027	3.0093

TABLE 5.15: Evaluation numbers with neighbour search (NS) and the fast neighbour search (FNS)

k	1	2	3	4	5
NS	72	2652	9143	60730	190634
FNS	21	136	480	960	2394

TABLE 5.16: Optimization objectives by neighbour search (NS) and the fast neighbour search (FNS)

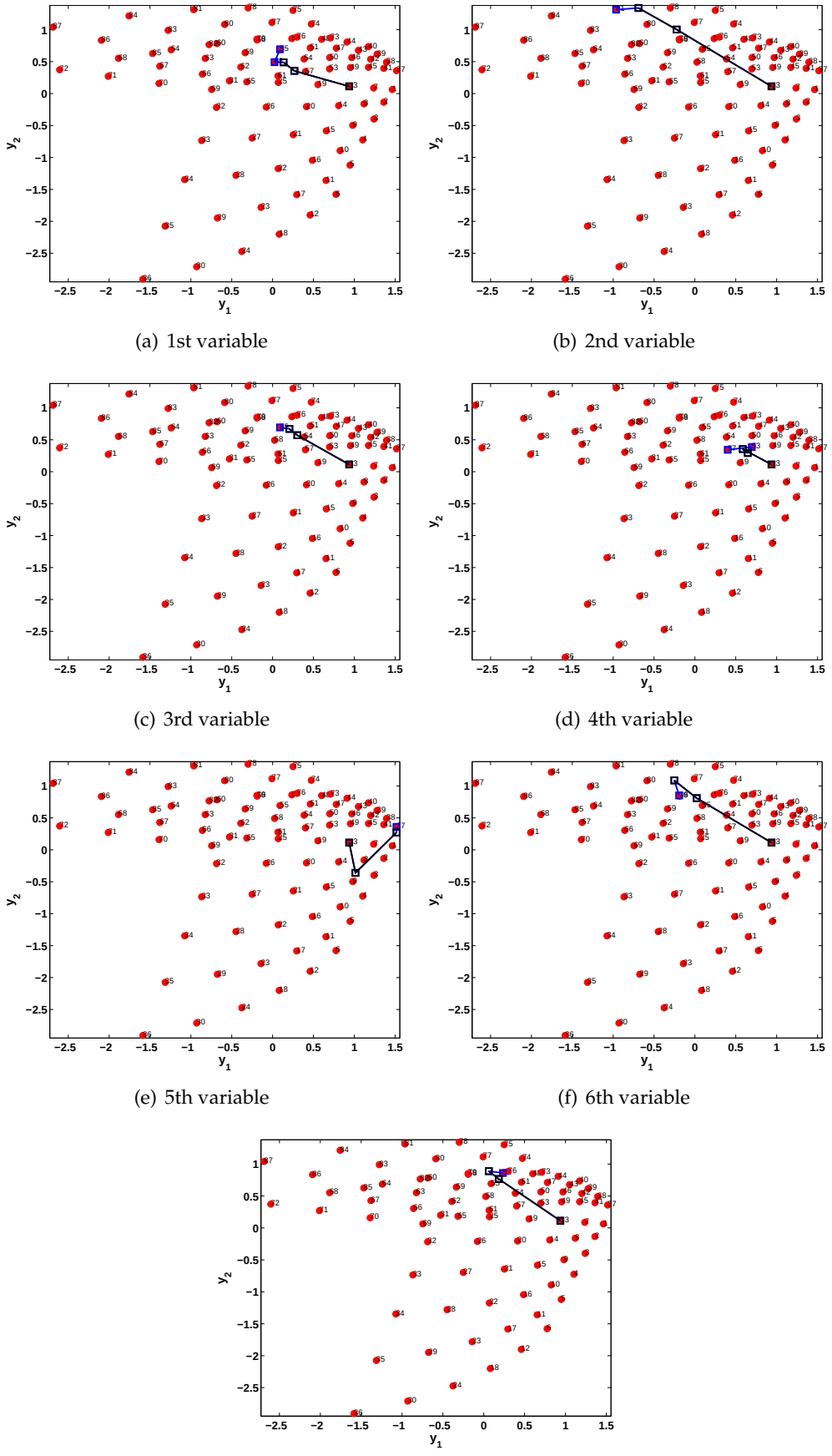
N_{nei}	4	5	6	7
NS	2.9954	2.9873	2.9874	2.9897
FNS	3.0106	3.0035	3.0027	3.0047

TABLE 5.17: Evaluation numbers with neighbour search (NS) and the fast neighbour search (FNS)

N_{nei}	4	5	6	7
NS	1318	2652	2737	3182
FNS	94	136	274	432

5.7 Conclusions and prospects

In this chapter, we propose a two-stage search strategy to handle categorical optimization problems: the continuous stage using MMA and the discrete stage using neighbour search, and the proposed strategy is applied in the ten-bar structure design problem and the dome structure optimization problem. We also test how the neighbour search parameters influence the optimization results. At last, we compare



(a) 1st variable

(b) 2nd variable

(c) 3rd variable

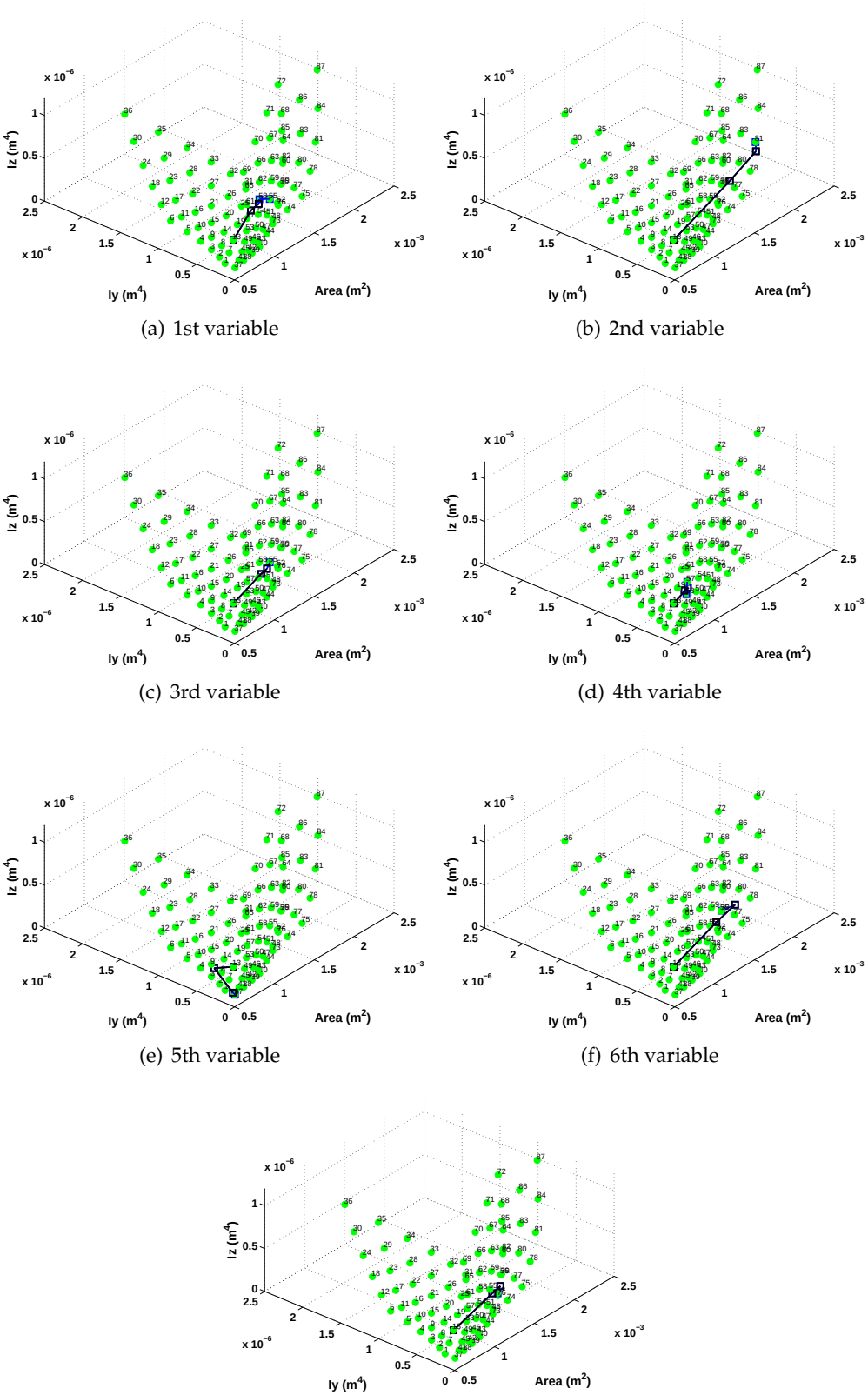
(d) 4th variable

(e) 5th variable

(f) 6th variable

(g) 7th variable

FIGURE 5.25: The evolution history of four variables in 2D space



(a) 1st variable (b) 2nd variable

(c) 3rd variable (d) 4th variable

(e) 5th variable (f) 6th variable

(g) 7th variable

FIGURE 5.26: The evolution history of four variables in the original 3D space

the proposed strategy with existing mature algorithms, including CMA-ES and SA. We can draw the conclusions that:
1) The two-stage search strategy is valid to obtain a k-order local solution.
2) The dimensionality reduction can help reduce the size of design space for the continuous search stage, thus leading to high optimization efficiency.
3) The number of search neighbours and search orders can affect the the optimization result in a similar manner: in the beginning, their increase can help obtain better solutions, but blindly increasing them cannot yield better results continuously.
4) The comparison between the two-stage optimization strategy, CMA-ES and SA shows the proposed MMA-MS has the highest optimization efficiency due to the continuous approximate search.
5) The fast neighbour search algorithm can significantly reduce the required evaluation numbers at a small cost of solution performance. So when the optimization problem scale is large, the fast neighbour search algorithm is suggested.
Further development of this proposed strategy will include introducing the k-manifolds learning algorithms to distinguish the design manifold piece by piece in the original design space.

Chapter 6

Categorical optimization: a k-manifolds learning approach

6.1 Background

In order to deal with categorical optimization problems, the discrete methods (Herrera, Lozano, and Verdegay, 1998; Goldberg, 2006) are widely used by encoding categorical variables with real numbers or binary strings. In previous chapters, we have already introduced two methods: the graph-based approach and the continuous-discrete two-stage search strategy. In this chapter, we will deal with a more complex situations in which the discrete design points (design space) are located on different manifolds and cannot be classified with one manifold learner. Moreover, the information on that which points belong to a given manifold is unknown. This is the main challenge we are facing.

Although both linear and non-linear manifold learning has achieved considerable progress, the research on multiple manifolds (k-manifolds) learning is still under exploration (Gong, Zhao, and Medioni, 2012). The multiple manifolds indicate that in the observation space, there is no single, global and integral manifold but are several local, separated or interacting manifolds. Despite of less research concern compared to manifold learning, there are also several helpful k-manifolds learning methods. For example the robust multiple manifolds structure learning (RMMSL) scheme (Gong, Zhao, and Medioni, 2012), hierarchical manifolds learning framework (Wang, Tiňo, and Fardal, 2008), Spectral Clustering (Wang et al., 2011) and joint-manifold method (Lee, Elgammal, and Torki, 2016). Those research works have stated the k-manifolds learning problems clearly, but none of them gives a general means to deal with all the k-manifolds learning problems.

As a linear manifold learning tool, principal component analysis (PCA) has won a large deal of attentions and developments. For example the Kernel PCA (Schölkopf, Smola, and Müller, 1998; Cao et al., 2003). The weighted principal component analysis (WPCA) (Cochran and Horne, 1977; Fan, Liu, and Xu, 2011) is also the variant of PCA. WPCA introduces weights to scale any a sample, dimension or element, thus influencing the final learning results. There are many publications on research and applications of WPCA, for example reconstruction of reflectance spectra (Agahian, Amirshahi, and Amirshahi, 2008), vendor selection (Petroni and Braglia, 2000), image processing (Cheng et al., 2011), fault detection and classification (Yue and Tomoyasu, 2004) et al. But few has applied WPCA in k-manifolds learning fields.

In this section, we propose to use WPCA to explore multiple manifolds by regarding the weights as the design variables. In section 6.2, we will explain the mechanism of k-manifolds learning. In section 6.3, three numerical tests are given. Section 6.4 presents the sectional conclusions and prospects.

6.2 K-manifolds learning

In this section, a new k-manifolds learning method based on weighted principal component analysis (WPCA) is proposed.

6.2.1 Weighted principal component analysis

Suppose the data is defined as $\mathbf{X} = [\mathbf{x}^1, \mathbf{x}^2, \cdots, \mathbf{x}^n]$ ($M \times n$). M is the number of dimensions, and n indicates the number of samples. In the weighted PCA, each sample $\mathbf{x}^j$ is attached with a weight w^j which controls the its contribution to the PCA process.
We write the weighted mean of the data as:

$$\mu = \frac{1}{\sum_{j=1}^{n} w^j} \sum_{j=1}^{n} w^j \mathbf{x}^j. \tag{6.1}$$

Then each point can be centralized by subtracting the weighted mean. Note that the data after centralization can be still labelled as $\mathbf{X}$ for the reason of consistency.
The covariance of the weighted data can be written as:

$$\mathbf{C} = \frac{1}{\sum_{j=1}^{n} w^j} \mathbf{X}\mathbf{W}\mathbf{X}^T, \tag{6.2}$$

where $\mathbf{W} = \mathrm{diag}(w^j)$.
By eigenvalue decomposition, we obtain:

$$\mathbf{C} = \mathbf{\Phi}\mathbf{\Lambda}\mathbf{\Phi}^T, \tag{6.3}$$

where $\mathbf{\Lambda}$ denotes the eigenvalues and $\mathbf{\Phi}$ is the metric of corresponding eigenvectors, and $\mathbf{\Phi}^T\mathbf{\Phi} = \mathbf{I}$. Note that after a descending ordering

$$\begin{aligned} &\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \lambda_2, \cdots, \lambda_M), \\ &\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_M. \end{aligned} \tag{6.4}$$

Note that all the eigenvectors are also sorted with the same order in Eq. (6.4).
We choose the first m eigenvectors and form the projection matrix $\mathbf{\Phi}_m = [\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \cdots, \boldsymbol{\phi}_m]$.
Finally, we obtain the reduced order representation by

$$\mathbf{y}^j = w^j \mathbf{\Phi}_m^T \mathbf{x}^j, \;\; j = 1, 2, \cdots, n. \tag{6.5}$$

With Eq. 3.4, we can project new points to the lower dimensional space.
Note that if the weight w^j equals to 1, it means the the corresponding data point $\mathbf{x}^j$ completely participates in the centralisation process and covariance matrix assembling processes. If w^j equals to 0, it indicates w^j neither affects the centralization nor plays a role in the covariance matrix. When $0 < w^j < 1$, it shows the point w^j takes an incomplete participation in the whole process.
The function of sample point weights help us with the idea that, if we set different 0/1 values to the data point weights, we will easily control which points are allowed to participate in PCA and which points cannot. Based on this idea, we develop an automatic way to determine the weights by using optimization means, in

order to distinguish different lower-dimensional manifolds in the higher-order extrinsic space.
We need to point out that, the proposed method can only be applied to the situations where the number of manifolds and the intrinsic dimensions of manifolds are already known or predefined in advance.

6.2.2 Optimization criteria

We suppose the number of manifolds s and the intrinsic dimensions of manifolds m are given. The manifolds are labelled as $\mathbf{Z}_1, \mathbf{Z}_2, \cdots, \mathbf{Z}_s$, respectively. w_k^j is the weight associated with data point $\mathbf{x}^j$ and manifold $\mathbf{Z}_k$. The physical meaning of w_k^j can be explained as the possibility by which the point $\mathbf{x}^j$ belongs to manifold $\mathbf{Z}_k$. We assembly all the weights together into weight matrix $\mathbf{W}$:

$$\mathbf{W} = \begin{bmatrix} w_1^1 & w_1^2 & \cdots & w_1^n \\ w_2^1 & w_2^2 & \cdots & w_2^n \\ \cdots & \cdots & \cdots & \cdots \\ w_s^1 & w_s^2 & \cdots & w_s^n \end{bmatrix}_{s\times n}. \tag{6.6}$$

The row of matrix $\mathbf{W}$ indicates the degree of the data belong to a certain manifold, and the column of matrix $\mathbf{W}$ means the effort of a certain data point makes. As a result, we need to find a suitable criteria to evaluate how those weights behave, namely to find the optimization objective.
We choose an arbitrary row of matrix $\mathbf{W}$ and mark it as $\mathbf{w} = [w_k^1 \quad w_k^2 \quad \cdots \quad w_k^n]$. Then WPCA is applied and yields the lower dimensional representation $\mathbf{Y}_k = (\mathbf{y}_k^1, \mathbf{y}_k^2, \cdots, \mathbf{y}_k^n)$ corresponding to the k-th row of $\mathbf{W}$.
Then we use the polynomial interpolation to rebuild higher dimensional data points $\mathbf{X}$. In order to distinguish the original data points and the reconstructed data points, we mark the constructed points with the star symbol. So we can write

$$(\mathbf{x}_k^{j^*})^T = \mathbf{p}^T(\mathbf{y}_k^j) \cdot \mathbf{D}, \;\; j = 1, 2, \cdots, n. \tag{6.7}$$

$\mathbf{p}^T(\mathbf{y}^j)$ is the polynomial expression and $\mathbf{D}$ is the coefficient matrix containing all the items of the polynomials.
For example, if the polynomial order is 2, M equals 3 and m equals 2, then $\mathbf{p}^T(\mathbf{y}_k^j)$ can be written as:

$$\mathbf{p}^T(\mathbf{y}_k^j) = (1, {}^1b^j, {}^2b^j, ({}^1b^j)^2, ({}^2b^j)^2, {}^1b^j \cdot {}^2b^j). \tag{6.8}$$

In order to obtain matrix $\mathbf{D}$, we firstly define the fitting difference corresponding to instance j and manifold k:

$$E_k^j = \| \; (\mathbf{p}^T(\mathbf{y}_k^j) \cdot \mathbf{D})^T - \mathbf{x}^j \; \|_2 \tag{6.9}$$

where $\| \cdot \|_2$ indicates the 2 norm of a vector.

The overall fitting error for all data points and all manifolds is written as:

$$\text{min.:} \quad E(\mathbf{W}) = \sum_{k=1}^{s}\sum_{j=1}^{n}(E_k^j)^2. \tag{6.10}$$

6.2.3 Optimization problem statement

Here we can write the discrete from of the k-manifold learning optimization problem:

$$\begin{aligned} &\text{min.:} \quad E(\mathbf{W}) \\ &\text{s. t.:} \quad \sum_{k=1}^{s} w_k^j = 1, \;\; j = 1, 2, \cdots, n; \\ &\qquad\quad w_k^j \in \{0, 1\}, \;\; j = 1, 2, \cdots, n; \;\; k = 1, 2, \cdots, s. \end{aligned} \tag{6.11}$$

The optimization statement in Eq. (6.11) involves large numbers of equality constraints. For example, when the genetic algorithm is used, the crossover and mutation operators will probably generate new populations which violate the equality constraints.
In order to solve this problem, we propose to use a different set of variables θ instead of w.
Here we write:

$$\begin{aligned} w_1^j &= cos^2(\theta_1^j)cos^2(\theta_2^j)\cdots cos^2(\theta_{s\text{-}1}^j); \\ w_2^j &= cos^2(\theta_1^j)cos^2(\theta_2^j)\cdots sin^2(\theta_{s\text{-}1}^j); \\ w_3^j &= cos^2(\theta_1^j)cos^2(\theta_2^j)\cdots sin^2(\theta_{s\text{-}2}^j); \\ &\cdots \\ w_{s\text{-}1}^j &= cos^2(\theta_1^j)sin^2(\theta_2^j); \\ w_s^j &= sin^2(\theta_1^j). \\ \theta_k^j &\in \{0, 0.5\pi\}, \;\; j = 1, 2, \cdots, n; \;\; k = 1, 2, \cdots, s. \end{aligned} \tag{6.12}$$

When the original variables w have been replaced by variables θ, the n equality constraints shown in Eq. (6.11) are naturally satisfied. Then the optimization problem can be expressed as:

$$\begin{aligned} &\text{min.:} \quad E(\mathbf{W}(\theta_k^j)) \\ &\text{s. t.:} \quad \theta_k^j \in \{0, 0.5\pi\}, \;\; j = 1, 2, \cdots, n; \;\; k = 1, 2, \cdots, s. \end{aligned} \tag{6.13}$$

Then the discrete optimization methods can be used to handle the problem (6.13).
We need to point out that continuous topology optimization methods can also be used to deal with the problem since we can make the design domain continuous as follows:

$$\begin{aligned} &\text{min.:} \quad E(\mathbf{W}(\theta_k^j)) \\ &\text{s. t.:} \quad 0 \le \theta_k^j \le 0.5\pi, \;\; j = 1, 2, \cdots, n; \;\; k = 1, 2, \cdots, s. \end{aligned} \tag{6.14}$$

When continuous topology optimization methods are applied, the penalization on the design variables are necessary for this penalization can push the design variables

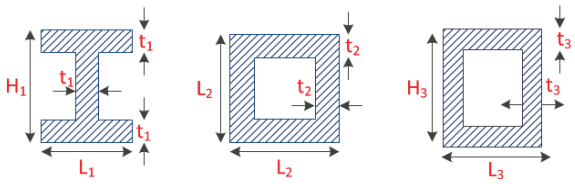


FIGURE 6.1: The three types of cross-sections

converge to 0 or 0.5π with the decreasing of the objective function.

6.2.4 Optimization algorithm for k-manifold learning

In our case, genetic algorithms are chosen to minimize the total fitting error by finding suitable combination of the weights w, or we say θ. In view of the particularity of the problem, some modifications to the genetic algorithms are necessary.
During the population initialization process, if we take random initialization in θ space, it is not hard to derive that the generating probability of w_s^j will be 0.5 ($j = 1, 2, \cdots, n$), and the other possibilities will share the other 0.5 possibility. This is caused by the introducing of θ variables, the sine and the cosine formulas. It can eliminate the equality constraints, meanwhile, it also has the drawback. The practical way to avoid that drawback is that, we execute the population initialization process in the w space, which can guarantee that the possibility of that any a point belongs to a given manifold is $\frac{1}{s}$. After the initialization in the w space, the populations are transformed into θ space for the following evaluation procedure.
In the crossover process, all θ variables attached to the same data point will be regarded as a solid group and cannot be split separately. Then the crossover operators only act on different groups of θ variables. This modification is designed to protect the ascription stability of any a data point. That means only the meaningful gene fragments are taken part in the crossover, instead of each gene unit.
The same measure is also taken to the mutation process. During the mutation, a gene fragment related to one data point will be entirely mutated to another possibility, rather than just make mutation gene by gene independently.

6.2.5 A numerical test

In our case, we have three types of cross-sections: the I-shape, the hollow square shape and the hollow rectangle shape, as illustrated in Fig. 6.1. The geometrical parameters of all the cross-sections are listed as in Tab. 6.1, respectively. Due to the varying of geometrical parameters, there are totally 40 instances for the I-shape cross-section, 49 instances for the hollow square cross-section and 16 instances for the hollow rectangle cross-section.

TABLE 6.1: The available values of geometrical parameters

Parameter	Available values(m)

H_1	$(0.05, 0.06, \cdots, 0.09)$
L_1	$(0.04, 0.05, \cdots, 0.11)$
t_1	(0.007)
L_2	$(0.06, 0.065, \cdots, 0.09)$
t_2	$(0.004, 0.001, \cdots, 0.01)$
H_3	$(0.065, 0.075, \cdots, 0.095)$
L_3	$(0.065, 0.075, \cdots, 0.095)$
t_3	(0.008)

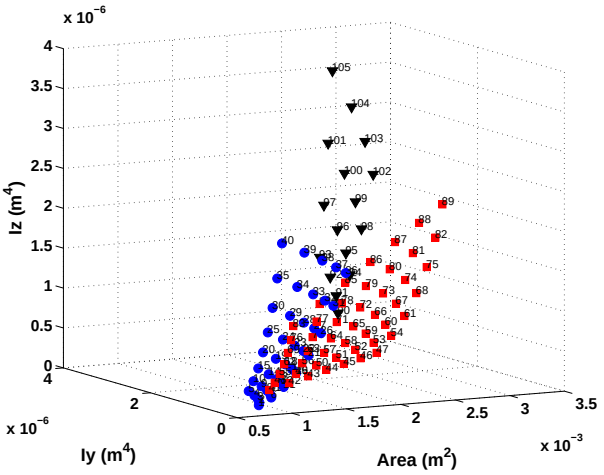


FIGURE 6.2: The physical properties of 105 cross-sections

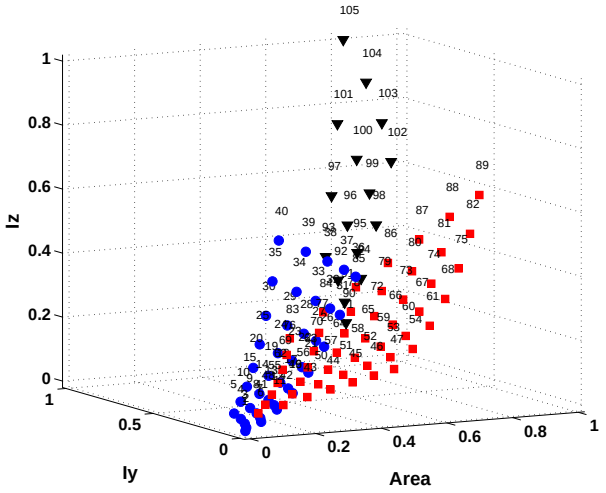


FIGURE 6.3: The normalized physical properties

The investigated physical properties include cross-section areas, area moments of inertia along y-axis and area moments of inertia along z-axis (x-axis is recognized as the normal direction of the area). We plot all 105 cross-section in a 3-d physical space, as shown in Fig. 6.2, then we normalize them in Fig. 6.3.

In the optimization, the population is 800, and the generation number is set as 100. The probabilities for crossover and mutation are 0.9 and 0.05, respectively. We set the manifold number as 3, and all the intrinsic dimensions of the manifolds are 2. Fig. 6.4 lists the decreasing history of the total fitting errors in 100 generations. And Fig. 6.5 shows the reduced order representation of 3 manifolds, respectively. We can find that although there are three data points which enter a wrong manifolds, showing that the optimization algorithm cannot find the global minima, the fitting errors are already small enough to execute further structural optimization. We also integrate the three 2-D manifolds into one figure, as shown in Fig. 6.6, through only translation and rotation operations. We need to note that after any translation or rotation operations, the coefficient matrix $\mathbf{D}$ needs to be uprated, but the total fitting errors will not change.

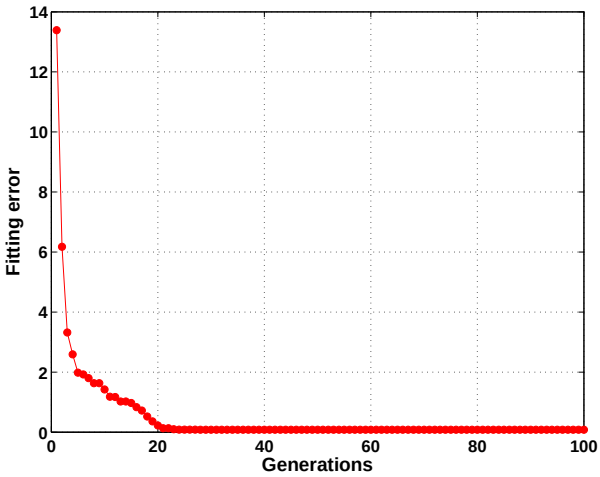
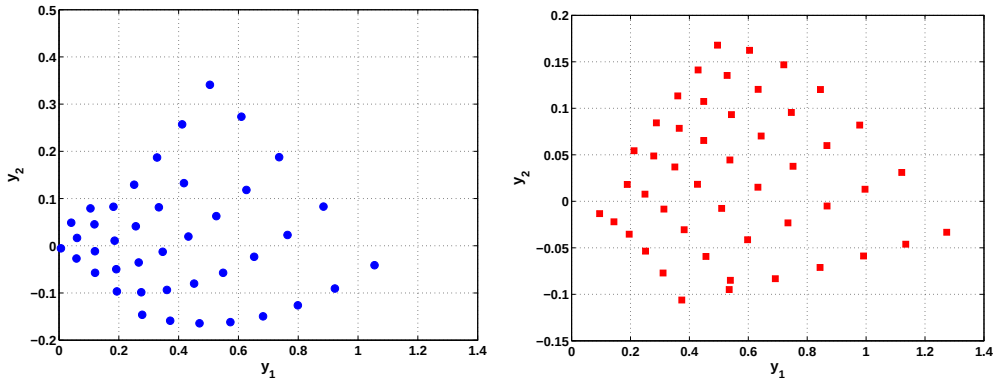
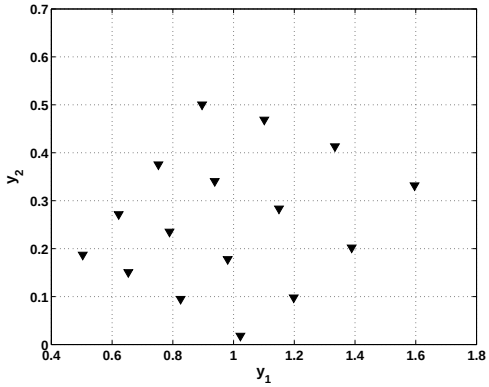


FIGURE 6.4: The objective: total errors in each iteration



(a) The 1st manifold

(b) The 2nd manifold



(c) The 3rd manifold

FIGURE 6.5: The final three manifolds

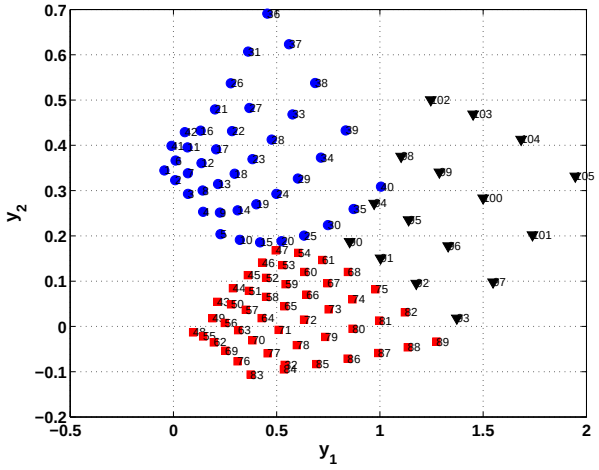


FIGURE 6.6: The integrated reduced representations of three manifolds

6.3 Numerical tests

Based on the re-organized and integrated lower dimensional graph, we utilize the two-stage search approach proposed in previous chapter to handle the optimization. We need to note that the junction area between two manifolds is discontinuous, thus an interpolation in the discontinuous area is always needed.

6.3.1 The 10-bar structure

As shown in Fig. 6.7, the ten-bar cantilever structure is as the same as that in previous chapter. The left end of the structure are fixed to rigid while the lower right end bears a downward 20000N force. The available cross-section types are shown in Fig. 6.2. We want to minimize the global strain energy with the functioning of the external load, while the mass of the whole structure should be lower than 700kg and linear buckling is not allowed. Here we write the expressions of critical forces for linear buckling:

$$f_y^{cr} = -\frac{\pi^2 EI_y}{L^2}, \quad f_z^{cr} = -\frac{\pi^2 EI_z}{L^2}. \tag{6.15}$$

We mark the critical buckling loads with negative signs for they can only happen under compression loads.
The optimization problem is formulated as:

$$\begin{aligned}
\text{min.:} \quad & Se(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4) = 0.5\mathbf{u}^T\mathbf{K}\mathbf{u}; \\
\text{s. t.:} \quad & \text{mass} - 700 \leq 0; \\
& \mathbf{K}\mathbf{u} = \mathbf{P}; \\
& \max(\mathbf{F}_y^{cr} - \mathbf{F}) \leq 0, \ \ \mathbf{F}_y^{cr} = (f_y^{cr1}, f_y^{cr2}, \cdots, f_y^{cr10}); \\
& \max(\mathbf{F}_z^{cr} - \mathbf{F}) \leq 0, \ \ \mathbf{F}_z^{cr} = (f_z^{cr1}, f_z^{cr2}, \cdots, f_z^{cr10}); \\
& \mathbf{x}_i \in \{\mathbf{x}_i^1, \mathbf{x}_i^2, \cdots, \mathbf{x}_i^{105}\}, \ \ i = 1, 2, 3, 4; \\
& \mathbf{x}_i^j = (A_i^j, I_{yi}^j, I_{zi}^j)^T, \ \ j = 1, 2, \cdots, 105.
\end{aligned} \tag{6.16}$$

where $\mathbf{F}$ denotes the row vector composed by the inner forces of all the bars.

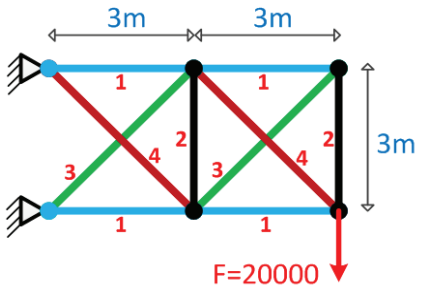


FIGURE 6.7: The ten-bar structure

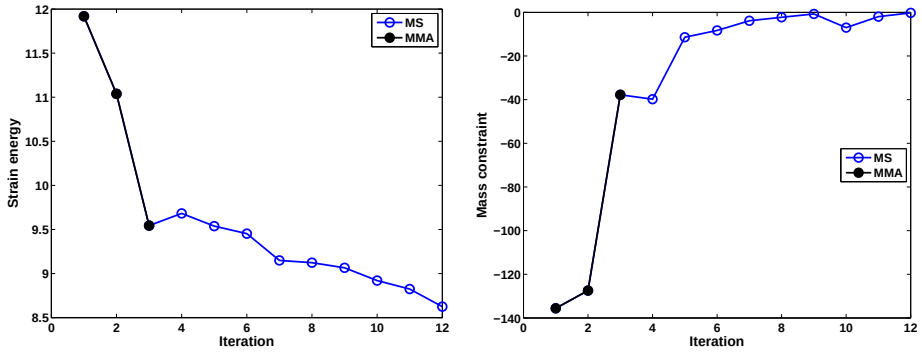
The initial choice for the four groups of bars are the bars with cross-section number 30, 36, 50 and 89. The neighbour number and the mutating variable number are set as 5 and 2, respectively. Fig. 6.8(a-c) show the design histories of the objective, the mass constraint and the two buckling constraints. In Fig. 6.8(a-c), the design histories with solid circle markers denote the continuous search process, while the following design histories with hollow circle markers mean the mutation search process. The final objective value is 8.6243J, with the mass constraint and buckling constraints all satisfied.
The evolution histories of designs are displayed in the reduced-order 2D space Fig. 6.9(a-d) and their corresponding original 3D space Fig. 6.10(a-d), respectively. Similar to the history figures of the response functions, optimization paths with solid circle markers indicate the continuous search stage, while the optimization paths with hollow circle markers denote the mutation search process. We can see that the k-manifolds learning method leads to the success of the ten bar structural optimization example.

6.3.2 The dam structure

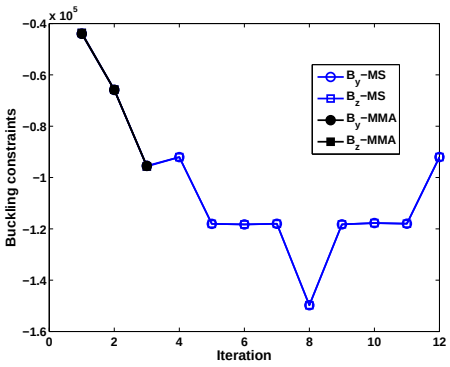
The Fig. 6.11 shows a dam structure which contains three groups of bars, and different group of bars are separated by different colors. The base of the dam structure is fixed on the ground while the right end of the structure are carrying four external forces 10000N, with force directions to the left. The material properties for all the bars are: Young's modulus E $= 2.1e11$Pa and the density $\rho = 7850\text{kg}/m^3$. The candidate cross-sections are also shown in Fig. 6.2. The optimization statement is written as follows:

$$
\begin{aligned}
\text{min.:}\quad & Se(\mathbf{x}_1,\mathbf{x}_2,\mathbf{x}_3) = 0.5\mathbf{u}^{\mathrm{T}}\mathbf{K}\mathbf{u};\\
\text{s. t.:}\quad & \text{mass} - 3000 \le 0;\\
& \mathbf{K}\mathbf{u} = \mathbf{P};\\
& \max(\mathbf{F}_y^{\mathrm{cr}} - \mathbf{F}) \le 0,\ \ \mathbf{F}_y^{\mathrm{cr}} = (f_y^{\mathrm{cr1}}, f_y^{\mathrm{cr2}}, \cdots, f_y^{\mathrm{cr105}});\\
& \max(\mathbf{F}_z^{\mathrm{cr}} - \mathbf{F}) \le 0,\ \ \mathbf{F}_z^{\mathrm{cr}} = (f_z^{\mathrm{cr1}}, f_z^{\mathrm{cr2}}, \cdots, f_z^{\mathrm{cr35}});\\
& \mathbf{x}_i \in \{\mathbf{x}_i^1, \mathbf{x}_i^2, \cdots, \mathbf{x}_i^{35}\},\ \ i = 1,2,3;\\
& \mathbf{x}_i^j = (A_i^j, I_{y_i}^j, I_{z_i}^j)^{\mathrm{T}},\ \ j = 1,2,\cdots,105.
\end{aligned}
\tag{6.17}
$$

The initial choice for the three groups of bars are cross-sections with number 36,

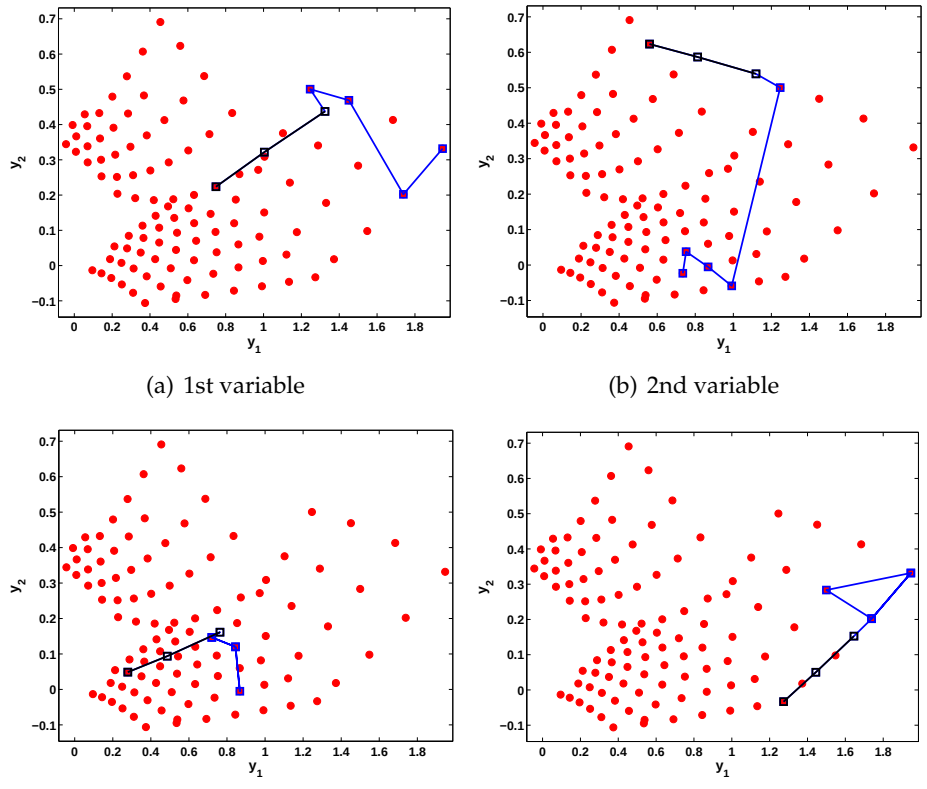


(a) The Objective (b) The mass constraint



(c) The two buckling constraints

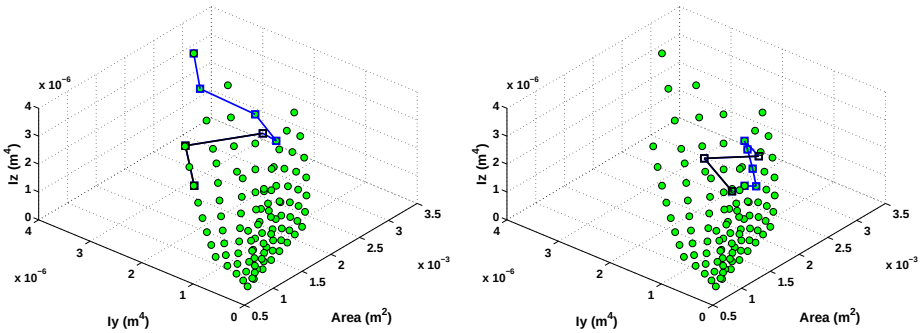
FIGURE 6.8: The design history of the objective and constraints



(a) 1st variable (b) 2nd variable

(c) 3rd variable (d) 4th variable

FIGURE 6.9: The evolution history of four variables in 2D space



(a) 1st variable (b) 2nd variable

(c) 3rd variable (d) 4th variable

FIGURE 6.10: The evolution history of four variables in the original 3D space

45 and 57. The neighbour number and the mutating variable number are set the same as previous test. Fig. 6.12(a-c) gives the optimization histories of the strain energy, the mass constraint and the two buckling constraints. The final objective value is 18.9408J, at the same time, the mass constraint and buckling constraints are all satisfied but only the mass constraint is active. Fig. 6.12(a-c), the design histories with solid circle markers denote the continuous search process, while the following design histories with hollow circle markers mean the mutation search process.
The evolution histories of designs are displayed in the reduced-order 2D space Fig. 6.13(a-d) and their corresponding original 3D space Fig. 6.14(a-d), respectively. It is also easy to obtain that the proposed k-manifolds learning method is valid for the given dame-like structural optimization problem.

6.3.3 The dome structure design

In the third test, the dome structure (Fig. 6.15) is the same as the example in section 5.5.2. The materials for each bar also remain the same as in the mentioned examples. The only difference with the example in section 5.5.2 is that the dome structure is carrying the vertical loads of 18kN, 9kN and 4.5kN. The objective and constraints

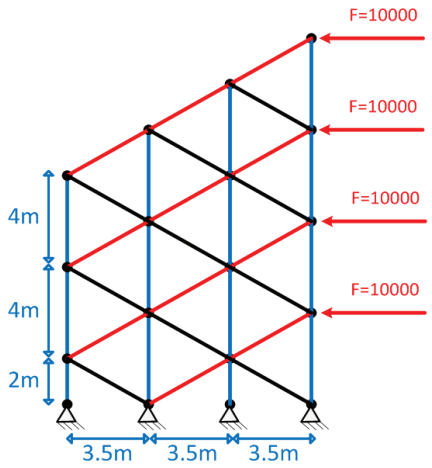
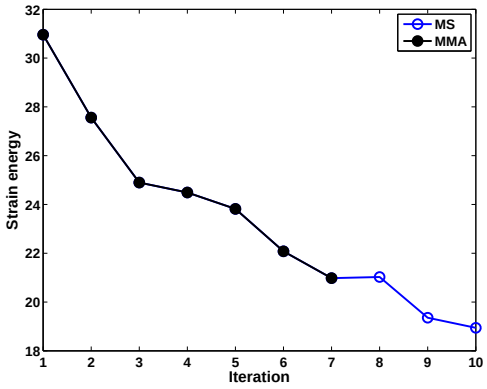
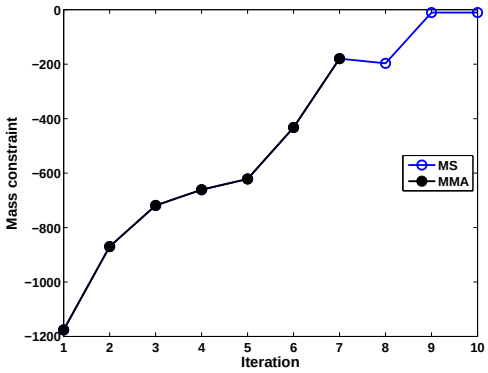


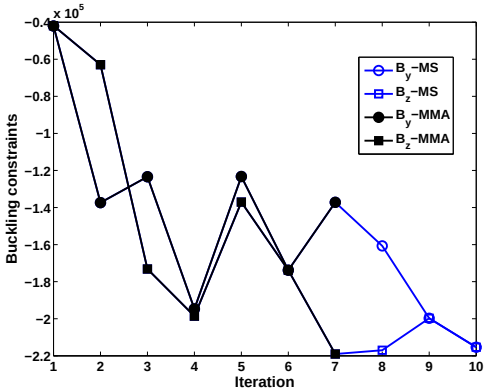
FIGURE 6.11: The dam structure



(a) The Objective

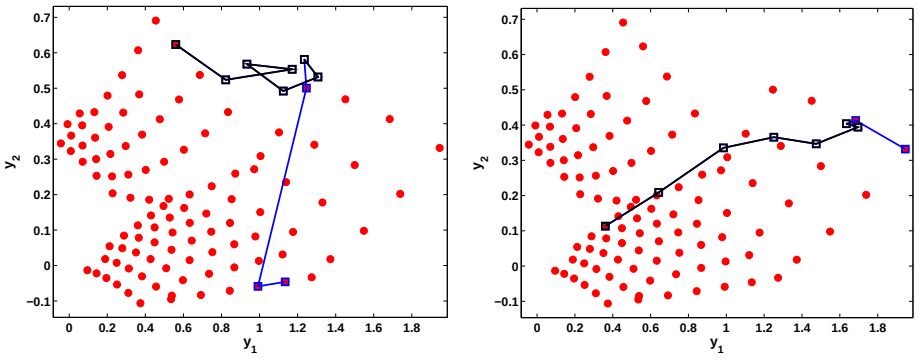


(b) The mass constraint

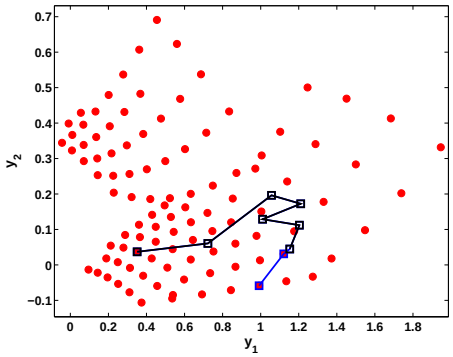


(c) The two buckling constraints

FIGURE 6.12: The design history of the objective and constraints

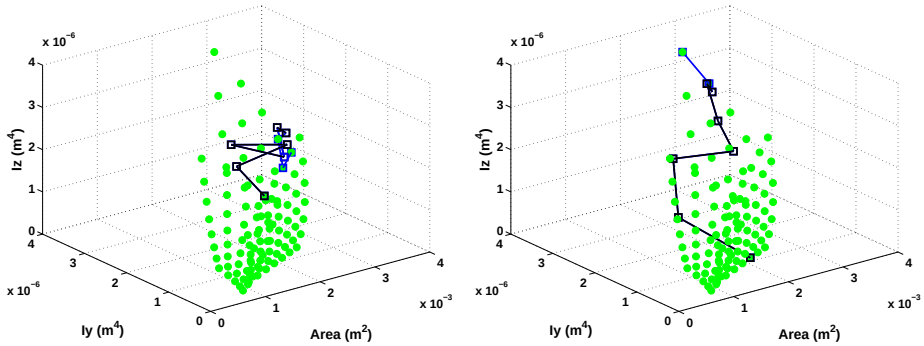


(a) 1st variable (b) 2nd variable

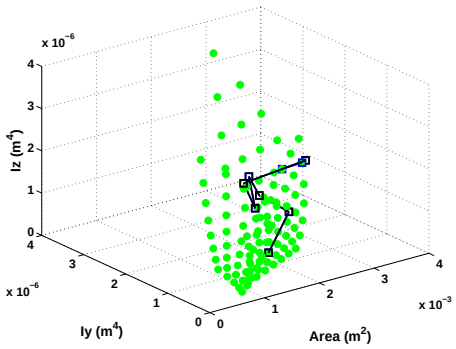


(c) 3rd variable

FIGURE 6.13: The evolution history of 3 variables in 2D space



(a) 1st variable (b) 2nd variable



(c) 3rd variable

FIGURE 6.14: The evolution history of 3 variables in the original 3D space

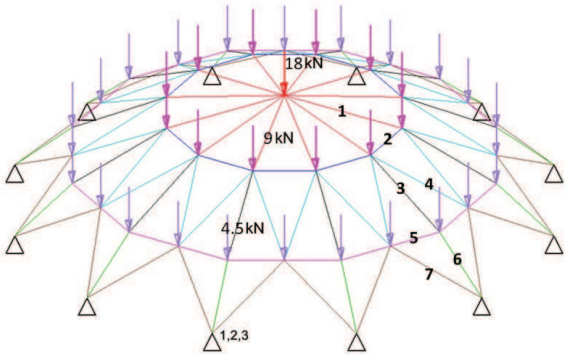


FIGURE 6.15: The dome structure

are also the same, shown as follows:

$$
\begin{aligned}
\text{min.:}\quad & Se(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_7) = 0.5\mathbf{u}^T\mathbf{K}\mathbf{u}; \\
\text{s. t.:}\quad & \text{mass} - 10000 \leq 0; \\
& \mathbf{K}\mathbf{u} = \mathbf{P}; \\
& \max(\mathbf{F}_y^{cr} - \mathbf{F}) \leq 0, \;\; \mathbf{F}_y^{cr} = (f_y^{cr1}, f_y^{cr2}, \cdots, f_y^{cr120}); \\
& \max(\mathbf{F}_z^{cr} - \mathbf{F}) \leq 0, \;\; \mathbf{F}_z^{cr} = (f_z^{cr1}, f_z^{cr2}, \cdots, f_z^{cr120}); \\
& \mathbf{x}_i \in \{\mathbf{x}_i^1, \mathbf{x}_i^2, \cdots, \mathbf{x}_i^{105}\}, \;\; i = 1, 2, \cdots, 7; \\
& \mathbf{x}_i^j = (A_i^j, I_{yi}^j, I_{zi}^j)^T, \;\; j = 1, 2, \cdots, 105.
\end{aligned}
\tag{6.18}
$$

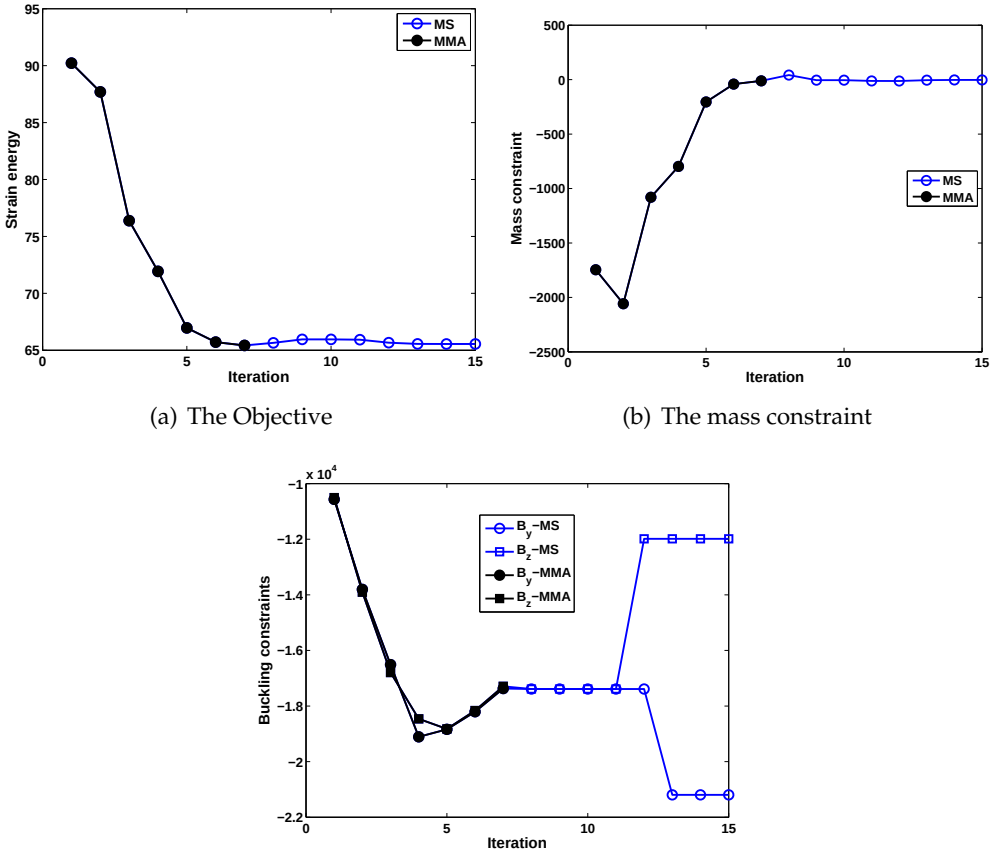
For the optimization, the mutation neighbour number and the mutated variable number are set to be 5 and 2, respectively. The initial design is (45 45 45 45 45 45 45). The final objective value is 65.5528J, as shown in Fig. 6.16(a). Both of the mass constraint and the buckling constraints are satisfied. The optimization paths in two kinds of design space are illustrated in Fig. 6.17 and Fig. 6.18, respectively. The results proves that the method is successful for this kind of problem.

6.4 Conclusions and prospects

In this section, we have proposed a k-manifolds learning method based on the weighted principal component analysis (WPCA), aiming at a deep simplification of the design space. Thus, it is used in the categorical structural optimization problems including the ten-bar truss design, the dame-like frame design and the dome structure design. All the three examples show that the method is valid to handle this kind of problems.

The proposed k-manifolds learning method also has several drawbacks, for example, it can only deal with linear or approximately linear manifolds; meanwhile, the number of potential manifolds and their intrinsic dimensions must be known; Also, the applied genetic algorithm cannot find the global solution when classifying different data points into different manifolds.

Further development may include a real non-linear weighted k-manifolds learning methods, which may based on the typical non-linear classifiers, for instance the Isomap, LLE or KPCA.

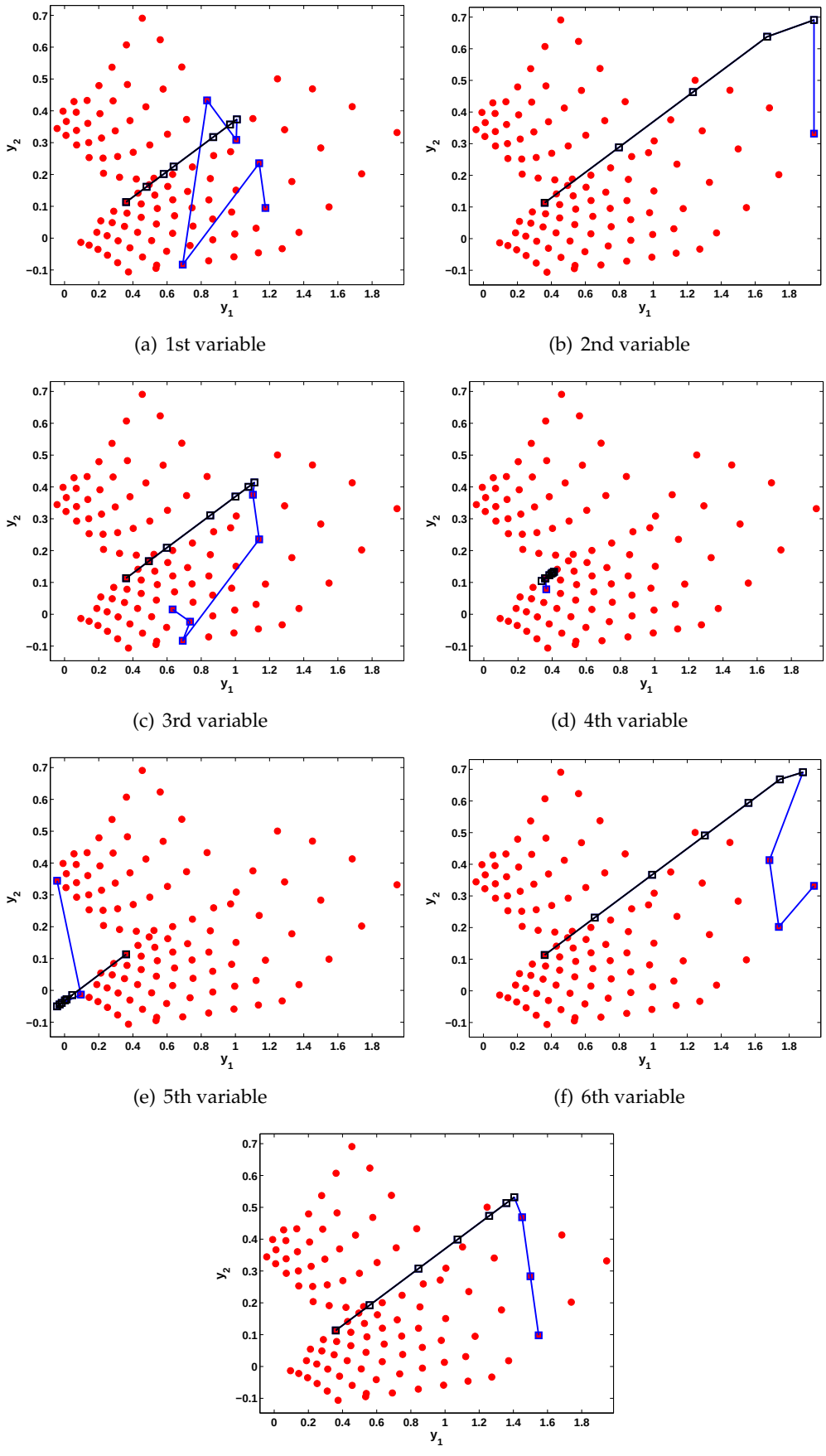


(a) The Objective

(b) The mass constraint

(c) The two buckling constraints

FIGURE 6.16: The design history of the objective and constraints



(a) 1st variable

(b) 2nd variable

(c) 3rd variable

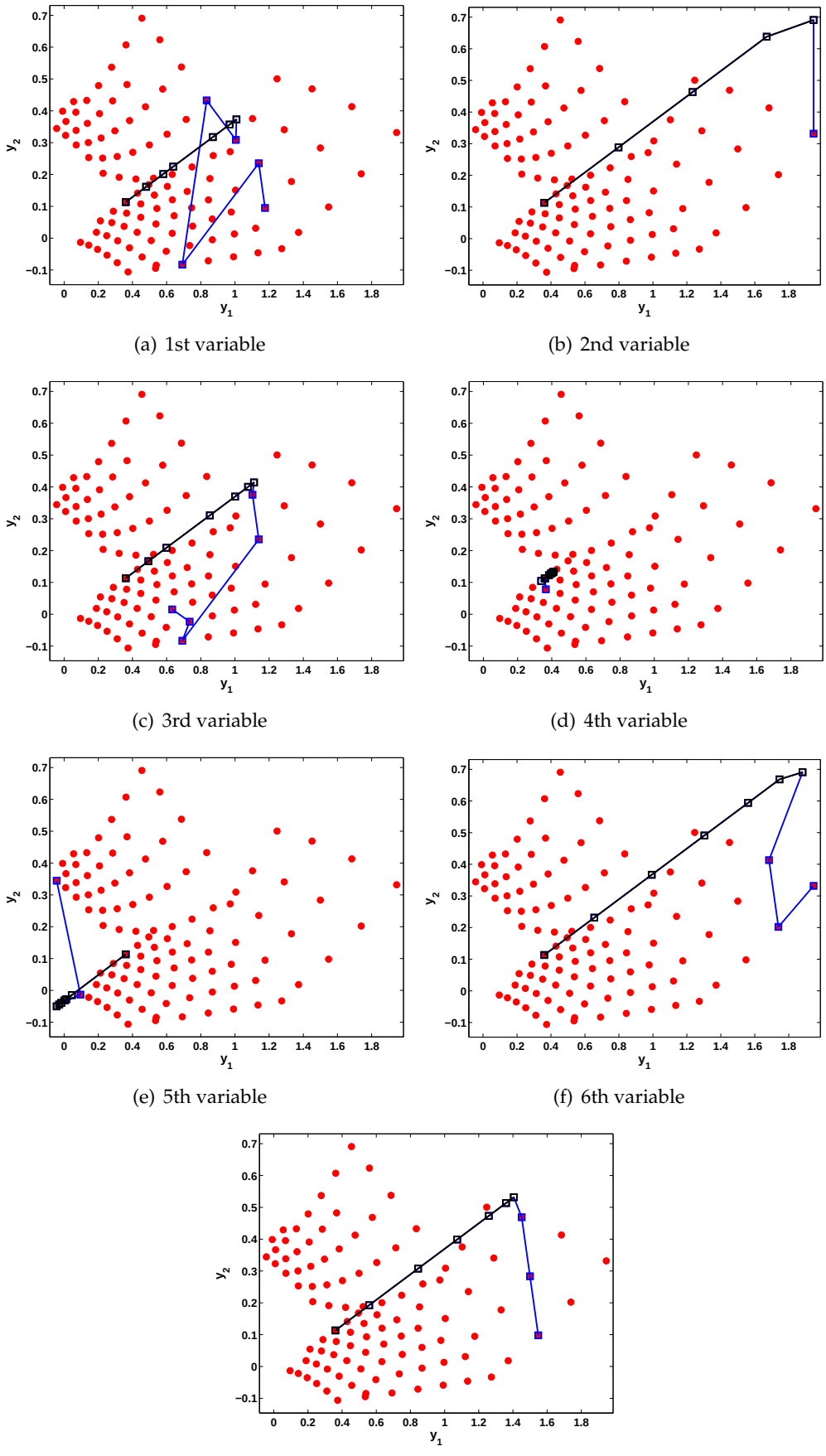
(d) 4th variable

(e) 5th variable

(f) 6th variable

(g) 7th variable

FIGURE 6.17: The evolution history of 7 variables in 2D space



(a) 1st variable

(b) 2nd variable

(c) 3rd variable

(d) 4th variable

(e) 5th variable

(f) 6th variable

(g) 7th variable

FIGURE 6.18: The evolution history of 7 variables in the original 3D space

Chapter 7

Conclusions, perspectives

7.1 Conclusions

In chapter 3, different manifold learning methods are discussed and compared in our work, the conclusions on manifold learning methods are drawn:
1) PCA and MDS can only deal with linear problems and some slightly non-linear problems. PCA and MDS will fail when dealing with strongly non-linear problems. For the efficiency, PCA performs faster than MDS.
2) As non-linear learner, Isomap and LLE can handle both linear and non-linear problems with proper learning parameters, for example the number of neighbours. Compared with LLE, Isomap can obtain the lower order mapping with less distortion, as we have illustrated in chapter 3, but the time cost of Isomap is normally two orders of magnitude higher than that of LLE. Consequently, for large scale manifold learning problems, for instance the problems with big sample numbers or high dimensions, LLE is suggested in priority.
3) Compared with other methods, KPCA is the most flexible one due to the margin of kernel functions. As a result, the learning results of KPCA also heavily depend on the choice of kernel functions. The second influencing factor is the parameters of corresponding kernel functions. We have also explained why KPCA cannot preserve point distances and angles with polynomial kernel functions (p>1).
In chapter 4, the proposed discrete evolutionary approach applies Isomap to reduce the dimensionality of the categorical design variables and form a lower dimensional graph based on which a discrete evolutionary algorithm is developed, including the crossover and mutation operators. This algorithm can help handle the multi-objective optimization problems and obtain the optimal Pareto sets with only admissible designs. Due to the application limitations of Isomap, this method can just deal with the case with one manifold.
In chapter 5, the two-stage search strategy includes the continuous stage using MMA and the discrete stage using neighbour search, and is applied into structural optimization problems. We can get the conclusions that:
1) Among those dimensionality reduction techniques, LLE has the least polynomial fitting errors for our case, compared with other manifold learning methods.
2) The two-stage search strategy has the highest optimization efficiency due to the continuous approximate search and the exquisite local mutation search, compared with other algorithms such as CMA-ES and SA.
3) The fast neighbour search algorithm can obtain comparable solutions with a significant reduction of function calls. It is suggested to handle large scale optimization problems.
In chapter 6, a k-manifolds learning method based on the weighted principal component analysis is proposed. Then the separated manifolds are classified taking advantages of the k-manifolds learning method. After the construction of the lower

order design space by polynomial fitting, the optimization method introduced in Chapter 5 is applied to structural optimization problems. The final optimization results demonstrate that the k-manifolds learning method is valid to deal with the design space of fragmented manifolds.

7.2 Perspectives

Further development of the research will include the exploration of non-linear k-manifolds learning methods to deal with the case in which the several non-linear manifolds are separated from each other. Another prospect is to discuss how to treat the optimization situations where the several manifolds have different intrinsic dimensionality. They will remain to be attractive research topics in the further work.

Appendix A

Questions frequently asked

- **What's the role of manifold learning to the whole work?**

The manifold learning plays an indispensable role during the development of methods for categorical optimization. The manifold learning reduces the dimensionality of categorical variables and maps the higher dimensional design space to a lower dimensional one. Based on the reduced order design space, the proposed optimization methods, including the discrete evolutionary approach (chapter 4) and the two-stage search approach (chapter 5), are developed. Even in chapter 6, the general optimization process is also as the same as that in chapter 4 and 5, despite that the k-manifolds learning methods takes the place of manifold learning.

- **What's the effect of manifold learning?**

We define the categorical variable as a kind of discrete multi-dimensional variable, but the different dimensions may be correlated with each other. In other words, the dimensions of categorical variables may be not independent.
The manifold learning can help reduce the dimensionality of categorical variables, leading to two advantages: firstly, the size of the problem is reduced, leading to lower computation cost and less computer storage in further handling of the problem, for example the sensitivity analysis; secondly, manifold learning reflects the real shape of the design space which helps designers to obtain a better understanding of the nature of the problem. The disadvantage includes that in some cases, manifold learning may result in a non-linearity of the design functions even if they are linear in the higher dimensional space.

- **How do you choose the optimization methods among the three proposed approaches?**

The three proposed search methods deal with different types of categorical optimization problems. For a multi-objective optimization problem with a single manifold design space, the discrete evolutionary approach is suggested. For a single-objective optimization problem with single manifold design space, the two-stage search method is the first choice. To handle single-objective optimization problems with several hidden manifolds, the k-manifolds learning based approach can be used. Noting that the number of constraint functions does not affect the choice of proposed methods, since all the methods can be applied with constrained or unconstrained optimization problems.

- **In chapter 3, why do you discuss distance and angle preservation with KPCA?**

Manifold learning maps the higher dimensional data to a lower dimensional space. However, the criteria to score different manifold learning methods as "good" or "bad" remains an open issue and has no thoroughly answers currently. Among all criteria,

the preservation of distance and angle is an important aspect. For example in the work (Gu et al., 2014), the researchers propose to use the distance and angle preservation as the general criteria to score and judge different manifold learning methods and can yield better learning results with all kinds of manifolds. This is the reason why we discuss distance and angle preservation with PCA in chapter 3.

- **Why do you use the total fitting error as the criteria to optimize and separate the different manifolds in higher dimensional space?**

The total fitting error is a measurement of the distance error between the current manifolds and real instance points. A smaller fitting error value indicates that the design manifolds are closer to the data points generally, which help improve the search accuracy in the continuous search stage. There are also other criteria to use in this steps. For example, one can utilize the distribution variance of the points corresponding to different manifolds as the judging criteria, but the approach may be affected by the non-linearity of the manifolds and has some limitations in real applications.

- **In section 4.5.1, you have 54 possible instances, but size of pop = 12, nb gen = 4, so you need 48 evaluations to optimize while an enumeration would require only 54 evaluations?**

Section 4.5.1 is a testing example to illustrate how the proposed approach works during the optimization process. In our point of view, enumeration is suitable for small-size problems, but may be invalid for large-scale ones. As categorical optimization problems are combination problems in nature, the designers are always desiring a more efficient way to handle this kind of problem instead of enumeration. Although the example in section 4.5.1 shows that the proposed approach performs with no big difference compared with enumeration, the following large-scale problems can prove the advantages of the proposed approach.

Appendix B

Attribute catalog of bars

TABLE 1: The attribute catalog of bar cross-sections for test 1 and test 2 in chapter 4

Section label	Area (m^2)	Iy (m^4)	Iz (m^4)
Section 1	2.827e-003	6.362e-007	6.362e-007
Section 2	3.421e-003	9.314e-007	9.314e-007
Section 3	4.072e-003	1.319e-006	1.319e-006
Section 4	4.778e-003	1.817e-006	1.817e-006
Section 5	5.542e-003	2.444e-006	2.444e-006
Section 6	6.362e-003	3.221e-006	3.221e-006
Section 7	7.238e-003	4.169e-006	4.169e-006
Section 8	8.171e-003	5.313e-006	5.313e-006
Section 9	9.161e-003	6.678e-006	6.678e-006
Section 10	3.600e-003	1.080e-006	1.080e-006
Section 11	4.356e-003	1.581e-006	1.581e-006
Section 12	5.184e-003	2.239e-006	2.239e-006
Section 13	6.084e-003	3.085e-006	3.085e-006
Section 14	7.056e-003	4.149e-006	4.149e-006
Section 15	8.100e-003	5.467e-006	5.467e-006
Section 16	9.216e-003	7.078e-006	7.078e-006
Section 17	3.125e-003	1.365e-006	1.365e-006

Section 18	5.000e-003	2.116e-006	2.116e-006
Section 19	3.750e-003	2.248e-006	2.248e-006
Section 20	6.250e-003	3.613e-006	3.613e-006
Section 21	4.375e-003	3.448e-006	3.448e-006
Section 22	7.500e-003	5.697e-006	5.697e-006
Section 23	5.000e-003	5.015e-006	5.015e-006
Section 24	8.750e-003	8.464e-006	8.464e-006
Section 25	3.456e-003	1.318e-006	1.318e-006
Section 26	6.283e-003	2.042e-006	2.042e-006
Section 27	4.084e-003	2.170e-006	2.170e-006
Section 28	7.540e-003	3.487e-006	3.487e-006
Section 29	4.712e-003	3.328e-006	3.328e-006
Section 30	8.796e-003	5.498e-006	5.498e-006
Section 31	5.175e-003	2.426e-006	2.053e-006
Section 32	5.589e-003	3.056e-006	2.217e-006
Section 33	6.003e-003	3.786e-006	2.382e-006
Section 34	6.417e-003	4.625e-006	2.546e-006
Section 35	6.300e-003	2.953e-006	3.704e-006
Section 36	6.804e-003	3.720e-006	4.001e-006
Section 37	7.308e-003	4.610e-006	4.297e-006
Section 38	7.812e-003	5.630e-006	4.593e-006
Section 39	2.500e-003	1.986e-006	8.870e-007
Section 40	4.375e-003	2.572e-006	1.790e-006
Section 41	2.812e-003	4.053e-006	8.911e-007
Section 42	5.000e-003	5.729e-006	1.823e-006

Section 43	3.125e-003	2.604e-006	2.091e-006
Section 44	5.625e-003	3.418e-006	4.199e-006
Section 45	3.438e-003	5.257e-006	2.096e-006
Section 46	6.250e-003	7.552e-006	4.232e-006
Section 47	4.400e-003	2.512e-006	2.512e-006
Section 48	5.000e-003	2.604e-006	2.604e-006
Section 49	4.900e-003	3.874e-006	2.906e-006
Section 50	5.625e-003	4.077e-006	3.027e-006
Section 51	4.900e-003	2.906e-006	3.874e-006
Section 52	5.625e-003	3.027e-006	4.077e-006
Section 53	5.400e-003	4.461e-006	4.461e-006
Section 54	6.250e-003	4.720e-006	4.720e-006

Appendix C

Attribute catalog of beams

TABLE 1: The attribute catalog of beam cross-sections for test 3 in chapter 4

Section label	Area (m^2)	Iy (m^4)	Iz (m^4)	Jx (m^4)
Section 1	2.124e-003	3.589e-007	3.589e-007	7.178e-007
Section 2	2.463e-003	4.827e-007	4.827e-007	9.655e-007
Section 3	2.827e-003	6.362e-007	6.362e-007	1.272e-006
Section 4	3.217e-003	8.235e-007	8.235e-007	1.647e-006
Section 5	3.632e-003	1.050e-006	1.050e-006	2.099e-006
Section 6	4.072e-003	1.319e-006	1.319e-006	2.638e-006
Section 7	1.936e-003	3.123e-007	3.123e-007	5.279e-007
Section 8	2.116e-003	3.731e-007	3.731e-007	6.306e-007
Section 9	2.304e-003	4.424e-007	4.424e-007	7.476e-007
Section 10	2.500e-003	5.208e-007	5.208e-007	8.802e-007
Section 11	2.704e-003	6.093e-007	6.093e-007	1.030e-006
Section 12	2.916e-003	7.086e-007	7.086e-007	1.198e-006
Section 13	3.136e-003	8.195e-007	8.195e-007	1.385e-006
Section 14	2.000e-003	5.592e-007	5.592e-007	1.250e-006
Section 15	2.700e-003	7.383e-007	7.383e-007	1.367e-006
Section 16	2.200e-003	7.250e-007	7.250e-007	1.664e-006
Section 17	3.000e-003	9.667e-007	9.667e-007	1.875e-006

Section 18	2.400e-003	9.208e-007	9.208e-007	2.160e-006
Section 19	3.300e-003	1.238e-006	1.238e-006	2.496e-006
Section 20	2.600e-003	1.149e-006	1.149e-006	2.746e-006
Section 21	3.600e-003	1.557e-006	1.557e-006	3.240e-006
Section 22	2.199e-003	3.436e-007	3.436e-007	6.872e-007
Section 23	3.063e-003	4.260e-007	4.260e-007	8.519e-007
Section 24	3.770e-003	4.712e-007	4.712e-007	9.425e-007
Section 25	2.513e-003	5.105e-007	5.105e-007	1.021e-006
Section 26	3.534e-003	6.461e-007	6.461e-007	1.292e-006
Section 27	4.398e-003	7.285e-007	7.285e-007	1.457e-006
Section 28	2.827e-003	7.245e-007	7.245e-007	1.449e-006
Section 29	4.006e-003	9.325e-007	9.325e-007	1.865e-006
Section 30	5.027e-003	1.068e-006	1.068e-006	2.136e-006
Section 31	1.750e-003	3.646e-007	1.786e-007	4.058e-007
Section 32	1.925e-003	4.853e-007	1.965e-007	4.752e-007
Section 33	2.100e-003	6.300e-007	2.144e-007	5.454e-007
Section 34	2.275e-003	8.010e-007	2.322e-007	6.160e-007
Section 35	2.250e-003	4.688e-007	3.797e-007	7.047e-007
Section 36	2.475e-003	6.239e-007	4.177e-007	8.417e-007
Section 37	2.700e-003	8.100e-007	4.556e-007	9.841e-007
Section 38	2.925e-003	1.030e-006	4.936e-007	1.130e-006
Section 39	1.950e-003	8.212e-007	3.209e-007	1.463e-007
Section 40	2.400e-003	8.800e-007	4.300e-007	3.200e-007
Section 41	2.100e-003	1.243e-006	3.238e-007	1.575e-007
Section 42	2.600e-003	1.362e-006	4.367e-007	3.467e-007

Section 43	2.250e-003	9.787e-007	5.484e-007	1.687e-007
Section 44	2.800e-003	1.053e-006	7.333e-007	3.733e-007
Section 45	2.400e-003	1.475e-006	5.512e-007	1.800e-007
Section 46	3.000e-003	1.625e-006	7.400e-007	4.000e-007
Section 47	2.064e-003	7.989e-007	5.723e-007	9.285e-007
Section 48	2.400e-003	8.550e-007	6.050e-007	9.302e-007
Section 49	2.304e-003	1.218e-006	6.618e-007	1.214e-006
Section 50	2.700e-003	1.323e-006	7.025e-007	1.235e-006
Section 51	2.304e-003	9.400e-007	9.400e-007	1.327e-006
Section 52	2.700e-003	1.012e-006	1.012e-006	1.367e-006
Section 53	2.544e-003	1.423e-006	1.081e-006	1.755e-006
Section 54	3.000e-003	1.555e-006	1.170e-006	1.838e-006

Bibliography

Abramson, Mark A et al. (2009). "Mesh adaptive direct search algorithms for mixed variable optimization". In: *Optimization Letters* 3.1, pp. 35–47.

Agahian, Farnaz, Seyed Ali Amirshahi, and Seyed Hossein Amirshahi (2008). "Reconstruction of reflectance spectra using weighted principal component analysis". In: *Color Research & Application* 33.5, pp. 360–371.

Aickelin, Uwe, Dipankar Dasgupta, and Feng Gu (2014). "Artificial immune systems". In: *Search methodologies*. Springer, pp. 187–211.

Allaire, Grégoire, François Jouve, and Anca-Maria Toader (2002). "A level-set method for shape optimization". In: *Comptes Rendus Mathematique* 334.12, pp. 1125–1130.

Altman, Edward I, Giancarlo Marco, and Franco Varetto (1994). "Corporate distress diagnosis: Comparisons using linear discriminant analysis and neural networks (the Italian experience)". In: *Journal of banking & finance* 18.3, pp. 505–529.

Anderson-Cook, Christine M (2005). *Practical genetic algorithms*.

Anzai, Yuichiro (2012). *Pattern recognition and machine learning*. Elsevier.

Ashby, MF (2000). "Multi-objective optimization in material design and selection". In: *Acta materialia* 48.1, pp. 359–369.

Ashby, Michael F and Kara Johnson (2013). *Materials and design: the art and science of material selection in product design*. Butterworth-Heinemann.

Back, Thomas (1996). *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press.

Balasubramanian, Mukund and Eric L Schwartz (2002). "The isomap algorithm and topological stability". In: *Science* 295.5552, pp. 7–7.

Banker, Rajiv D and Richard C Morey (1986). "The use of categorical variables in data envelopment analysis". In: *Management science* 32.12, pp. 1613–1627.

Barlow, Richard E and Frank Proschan (1975). *Statistical theory of reliability and life testing: probability models*. Tech. rep. Florida State Univ Tallahassee.

Bendsøe, Martin P (1989). "Optimal shape design as a material distribution problem". In: *Structural optimization* 1.4, pp. 193–202.

Bendsøe, Martin Philip and Noboru Kikuchi (1988). "Generating optimal topologies in structural design using a homogenization method". In: *Computer methods in applied mechanics and engineering* 71.2, pp. 197–224.

Bendsoe, Martin Philip and Ole Sigmund (2013). *Topology optimization: theory, methods, and applications*. Springer Science & Business Media.

Biegler, Lorenz T (1984). "Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation". In: *Computers & chemical engineering* 8.3-4, pp. 243–247.

Bishop, Christopher M, Markus Svensén, and Christopher KI Williams (1998). "GTM: The generative topographic mapping". In: *Neural computation* 10.1, pp. 215–234.

Boggs, Paul T and Jon W Tolle (1995). "Sequential quadratic programming". In: *Acta numerica* 4, pp. 1–51.

Branke, Jürgen et al. (2000). "A multi-population approach to dynamic optimization problems". In: *Evolutionary Design and Manufacture*. Springer, pp. 299–307.

Broyden, Charles G (1970). "The convergence of a class of double-rank minimization algorithms: 2. The new algorithm". In: *IMA Journal of Applied Mathematics* 6.3, pp. 222–231.

Cao, LJ et al. (2003). "A comparison of PCA, KPCA and ICA for dimensionality reduction in support vector machine". In: *Neurocomputing* 55.1, pp. 321–336.

Chang, Yin-Wen et al. (2010). "Training and testing low-degree polynomial data mappings via linear SVM". In: *Journal of Machine Learning Research* 11.Apr, pp. 1471–1490.

Chen, XB and Michael M Kostreva (1999). "A generalization of the norm-relaxed method of feasible directions". In: *Applied Mathematics and Computation* 102.2-3, pp. 257–272.

Cheng, GD and Xiao Guo (1997). "ε-relaxed approach in structural topology optimization". In: *Structural optimization* 13.4, pp. 258–266.

Cheng, Qiuming et al. (2011). "A spatially weighted principal component analysis for multi-element geochemical data for mapping locations of felsic intrusions in the Gejiu mineral district of Yunnan, China". In: *Computers & Geosciences* 37.5, pp. 662–669.

Chun, Jang-Sung et al. (1997). "Shape optimization of electromagnetic devices using immune algorithm". In: *IEEE transactions on magnetics* 33.2, pp. 1876–1879.

Cochran, Robert N and Frederick H Horne (1977). "Statistically weighted principal component analysis of rapid scanning wavelength kinetics experiments". In: *Analytical Chemistry* 49.6, pp. 846–853.

Coelho, Rajan Filomeno et al. (2015). "Investigation of Three Genotypes for Mixed Variable Evolutionary Optimization". In: pp. 309–319.

Coello, Carlos A Coello, David A Van Veldhuizen, and Gary B Lamont (2002). *Evolutionary algorithms for solving multi-objective problems*. Vol. 242. Springer.

Cortes, Corinna and Vladimir Vapnik (1995). "Support vector machine". In: *Machine learning* 20.3, pp. 273–297.

Csébfalvi, Anikó (2013). *ANGEL: A Simplified Hybrid Metaheuristic for Structural Optimization*. INTECH Open Access Publisher.

Davis, Lawrence (1991). "Handbook of genetic algorithms". In:

De Ridder, Dick et al. (2003). "Supervised locally linear embedding". In: *ICANN*. Springer, pp. 333–341.

Deb, Kalyanmoy et al. (2002). "Scalable multi-objective optimization test problems". In: *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*. Vol. 1. IEEE, pp. 825–830.

Dennis Jr, John E and Robert B Schnabel (1996). *Numerical methods for unconstrained optimization and nonlinear equations*. Vol. 16. Siam.

Deo, Narsingh, Janusz S Kowalik, et al. (1983). *Discrete optimization algorithms: with Pascal programs*. Courier Corporation.

Devroye, Luc and Terry J Wagner (1982). "8 Nearest neighbor methods in discrimination". In: *Handbook of Statistics* 2, pp. 193–197.

Dijkstra, Edsger W (1959). "A note on two problems in connexion with graphs". In: *Numerische mathematik* 1.1, pp. 269–271.

Domingos, Pedro and Michael Pazzani (1997). "On the optimality of the simple Bayesian classifier under zero-one loss". In: *Machine learning* 29.2-3, pp. 103–130.

Donoho, David L and Carrie Grimes (2003). "Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data". In: *Proceedings of the National Academy of Sciences* 100.10, pp. 5591–5596.

Dorigo, Marco and Mauro Birattari (2011). "Ant colony optimization". In: *Encyclopedia of machine learning*. Springer, pp. 36–39.

Dorigo, Marco, Vittorio Maniezzo, and Alberto Colorni (1996). "Ant system: optimization by a colony of cooperating agents". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 26.1, pp. 29–41.

Duysinx, Pierre and Martin P Bendsøe (1998). "Topology optimization of continuum structures with local stress constraints". In: *International journal for numerical methods in engineering* 43.8, pp. 1453–1478.

Eshelman, Larry J and J David Schaffer (1993). "Real-coded genetic algorithms and interval-schemata". In: *Foundations of genetic algorithms*. Vol. 2. Elsevier, pp. 187–202.

Etman, LFP et al. (1996). "Crash worthiness design optimization using multipoint sequential linear programming". In: *Structural Optimization* 12.4, pp. 222–228.

Fan, Zizhu, Ergen Liu, and Baogen Xu (2011). "Weighted principal component analysis". In: *International Conference on Artificial Intelligence and Computational Intelligence*. Springer, pp. 569–574.

Feo, Thomas A and Mauricio GC Resende (1995). "Greedy randomized adaptive search procedures". In: *Journal of global optimization* 6.2, pp. 109–133.

Ferreira, António JM (2008). *MATLAB codes for finite element analysis: solids and structures*. Vol. 157. Springer Science & Business Media.

Fessler, Jeffrey A and Alfred O Hero (1994). "Space-alternating generalized expectation-maximization algorithm". In: *IEEE Transactions on Signal Processing* 42.10, pp. 2664–2677.

Filomeno Coelho, Rajan (2012). "Extending moving least squares to mixed variables for metamodel-assisted optimization". In:

— (2014). "Metamodels for mixed variables based on moving least squares: Application to the structural analysis of a rigid frame". In: *Optimization and engineering* 15.2, pp. 311–329.

Fogel, Lawrence J (1999). *Intelligence through simulated evolution: forty years of evolutionary programming*. John Wiley & Sons, Inc.

Fogel, Lawrence J, Alvin J Owens, and Michael J Walsh (1966). "Artificial intelligence through simulated evolution". In:

Fu, Yangguang, Mingyue Ding, and Chengping Zhou (2012). "Phase angle-encoded and quantum-behaved particle swarm optimization applied to three-dimensional route planning for UAV". In: *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 42.2, pp. 511–526.

Fu, Yun, Shuicheng Yan, and Thomas S Huang (2008). "Classification and feature extraction by simplexization". In: *IEEE Transactions on Information Forensics and Security* 3.1, pp. 91–100.

Furey, Terrence S et al. (2000). "Support vector machine classification and validation of cancer tissue samples using microarray expression data". In: *Bioinformatics* 16.10, pp. 906–914.

Gao, Tong and Weihong Zhang (2011). "A mass constraint formulation for structural topology optimization with multiphase materials". In: *International Journal for Numerical Methods in Engineering* 88.8, pp. 774–796.

Goldberg, David E (2006). *Genetic algorithms*. Pearson Education India.

Goldberg, David E and John H Holland (1988). "Genetic algorithms and machine learning". In: *Machine learning* 3.2, pp. 95–99.

Goldstein, Allen A (1965). "On steepest descent". In: *Journal of the Society for Industrial and Applied Mathematics, Series A: Control* 3.1, pp. 147–151.

Gong, Dian, Xuemei Zhao, and Gérard Medioni (2012). "Robust multiple manifolds structure learning". In: *arXiv preprint arXiv:1206.4624*.

Gu, Yanchun et al. (2014). "A Manifold Learning Fusion Algorithm Based on Distance and Angle Preservation". In: *Pattern Recognition*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 31–43. ISBN: 978-3-662-45646-0.

Hansen, Nikolaus, Sibylle D Müller, and Petros Koumoutsakos (2003). "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)". In: *Evolutionary computation* 11.1, pp. 1–18.

Hartigan, John A (1975). "Clustering algorithms". In:

Hartigan, John A and Manchek A Wong (1979). "Algorithm AS 136: A k-means clustering algorithm". In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 28.1, pp. 100–108.

Herrera, Francisco, Manuel Lozano, and Jose L. Verdegay (1998). "Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis". In: *Artificial intelligence review* 12.4, pp. 265–319.

Herrera, Manuel et al. (2014). "Metamodel-assisted optimization based on multiple kernel regression for mixed variables". In: *Structural and Multidisciplinary Optimization* 49.6, pp. 979–991.

Hilbert, Renan et al. (2006). "Multi-objective shape optimization of a heat exchanger using parallel genetic algorithms". In: *International Journal of Heat and Mass Transfer* 49.15-16, pp. 2567–2577.

Ho, Shinn-Ying et al. (2008). "OPSO: Orthogonal particle swarm optimization and its application to task assignment problems". In: *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 38.2, pp. 288–298.

Houck, Christopher R, Jeff Joines, and Michael G Kay (1995). "A genetic algorithm for function optimization: a Matlab implementation". In: *NCSU-IE TR* 95.09.

Jain, Anil K and Richard C Dubes (1988). "Algorithms for clustering data". In:

Janis, Christine (1976). "The evolutionary strategy of the Equidae and the origins of rumen and cecal digestion". In: *Evolution* 30.4, pp. 757–774.

Jolliffe, Ian (2002). *Principal component analysis*. Wiley Online Library.

Jolliffe, Ian T (1986). "Principal component analysis and factor analysis". In: *Principal component analysis*. Springer, pp. 115–128.

Kandogan, Eser (2001). "Visualizing multi-dimensional clusters, trends, and outliers using star coordinates". In: *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pp. 107–116.

Kaufmann, Markus, Dan Zenkert, and Per Wennhage (2010). "Integrated cost/weight optimization of aircraft structures". In: *Structural and Multidisciplinary Optimization* 41.2, pp. 325–334.

Kaveh, A and S Talatahari (2009). "Size optimization of space trusses using Big Bang–Big Crunch algorithm". In: *Computers & structures* 87.17, pp. 1129–1140.

Kaveh, A and A Zolghadr (2011). "Shape and size optimization of truss structures with frequency constraints using enhanced charged system search algorithm". In:

Kennedy, James (2011). "Particle swarm optimization". In: *Encyclopedia of machine learning*. Springer, pp. 760–766.

Kirkpatrick, Scott, C Daniel Gelatt, and Mario P Vecchi (1983). "Optimization by simulated annealing". In: *science* 220.4598, pp. 671–680.

Kokkolaras, Michael, Charles Audet, and John E Dennis (2001). "Mixed variable optimization of the number and composition of heat intercepts in a thermal insulation system". In: *Optimization and Engineering* 2.1, pp. 5–29.

Kort, Barry W and Dimitri P Bertsekas (1972). "A new penalty function method for constrained minimization". In: *Decision and Control, 1972 and 11th Symposium on Adaptive Processes. Proceedings of the 1972 IEEE Conference on*. IEEE, pp. 162–166.

Koza, John R (1992). *Genetic Programming II, Automatic Discovery of Reusable Subprograms*. MIT Press, Cambridge, MA.

Lake, Brenden M, Ruslan Salakhutdinov, and Joshua B Tenenbaum (2015). "Human-level concept learning through probabilistic program induction". In: *Science* 350.6266, pp. 1332–1338.

Laporta, Paolo and Marcello Brussard (1991). "Design criteria for mode size optimization in diode-pumped solid-state lasers". In: *IEEE Journal of Quantum Electronics* 27.10, pp. 2319–2326.

Lasdon, Leon S et al. (1978). "Design and testing of a generalized reduced gradient code for nonlinear programming". In: *ACM Transactions on Mathematical Software (TOMS)* 4.1, pp. 34–50.

Lee, Chan-Su, Ahmed Elgammal, and Marwan Torki (2016). "Learning representations from multiple manifolds". In: *Pattern Recognition* 50, pp. 74–87.

Lee, Ki-Don and Kwang-Yong Kim (2010a). "Shape optimization of a fan-shaped hole to enhance film-cooling effectiveness". In: *International Journal of Heat and Mass Transfer* 53.15-16, pp. 2996–3005.

Lee, Namgil and Jong-Min Kim (2010b). "Conversion of categorical variables into numerical variables via Bayesian network classifiers for binary classifications". In: *Computational Statistics & Data Analysis* 54.5, pp. 1247–1265.

Li, Qing et al. (1999). "Shape and topology design for heat conduction by evolutionary structural optimization". In: *International Journal of Heat and Mass Transfer* 42.17, pp. 3361–3371.

Liang, JJ et al. (2014). "Problem definitions and evaluation criteria for the CEC 2015 competition on learning-based real-parameter single objective optimization". In: *Technical Report201411A, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore*.

Liao, Tianjun et al. (2014). "Ant colony optimization for mixed-variable optimization problems". In: *IEEE Transactions on Evolutionary Computation* 18.4, pp. 503–518.

Lindroth, Peter and Michael Patriksson (2011). *Pure Categorical Optimization: A Global Descent Approach*. Department of Mathematical Sciences, Chalmers University of Technology, University of Gothenburg.

Loke, Meng Heng and RD Barker (1996). "Rapid least-squares inversion of apparent resistivity pseudosections by a quasi-Newton method". In: *Geophysical prospecting* 44.1, pp. 131–152.

Lund, Erik (2009). "Buckling topology optimization of laminated multi-material composite shell structures". In: *Composite Structures* 91.2, pp. 158–167.

Lunga, Dalton et al. (2014). "Manifold-learning-based feature extraction for classification of hyperspectral data: A review of advances in manifold learning". In: *IEEE Signal Processing Magazine* 31.1, pp. 55–66.

Marcotte, Patrice and Jean-Pierre Dussault (1989). "A sequential linear programming algorithm for solving monotone variational inequalities". In: *SIAM Journal on Control and Optimization* 27.6, pp. 1260–1278.

Martin, Ingrid M and Sevgin Eroglu (1993). "Measuring a multi-dimensional construct: country image". In: *Journal of business research* 28.3, pp. 191–210.

McCane, Brendan and Michael Albert (2008). "Distance functions for categorical and mixed variables". In: *Pattern Recognition Letters* 29.7, pp. 986–993.

Meng, Liang et al. (2015). "Identification of material properties using indentation test and shape manifold learning approach". In: *Computer Methods in Applied Mechanics and Engineering* 297, pp. 239–257.

Meng, Liang et al. (2016). "Nonlinear Shape-Manifold Learning Approach: Concepts, Tools and Applications". In: *Archives of Computational Methods in Engineering*, pp. 1–21.

Michalewicz, Zbigniew and Girish Nazhiyath (1995). "Genocop III: A co-evolutionary algorithm for numerical optimization problems with nonlinear constraints". In: *Evolutionary Computation, 1995., IEEE International Conference on*. Vol. 2. IEEE, pp. 647–651.

Mika, Sebastian et al. (1999). "Fisher discriminant analysis with kernels". In: *Neural networks for signal processing IX, 1999. Proceedings of the 1999 IEEE signal processing society workshop.* Ieee, pp. 41–48.

Minyi, Yue and Han Jiye (1979). "A new reduced gradient method". In: *Scientia Sinica* 22.10, pp. 1099–1113.

Moon, Todd K (1996). "The expectation-maximization algorithm". In: *IEEE Signal processing magazine* 13.6, pp. 47–60.

Moscato, Pablo et al. (1989). "On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms". In: *Caltech concurrent computation program, C3P Report* 826, p. 1989.

Mosselman, Sietse, Jan Polman, and Rein Dijkema (1996). "ERβ: identification and characterization of a novel human estrogen receptor". In: *FEBS letters* 392.1, pp. 49–53.

Nasrabadi, Nasser M (2007). "Pattern recognition and machine learning". In: *Journal of electronic imaging* 16.4, p. 049901.

Nemec, Marian, David W Zingg, and Thomas H Pulliam (2004). "Multipoint and multi-objective aerodynamic shape optimization". In: *AIAA journal* 42.6, pp. 1057–1065.

Nocedal, Jorge and Stephen J Wright (2006). *Sequential quadratic programming*. Springer.

Pan, Quan-Ke et al. (2010). "A self-adaptive global best harmony search algorithm for continuous optimization problems". In: *Applied Mathematics and Computation* 216.3, pp. 830–848.

Patwari, Neal and Alfred O Hero (2004). "Manifold learning algorithms for localization in wireless sensor networks". In: *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on*. Vol. 3. IEEE, pp. iii–857.

Patwari, Neal, Alfred O Hero III, and Adam Pacholski (2005). "Manifold learning visualization of network traffic data". In: *Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data*. ACM, pp. 191–196.

Petroni, Alberto and Marcello Braglia (2000). "Vendor selection using principal component analysis". In: *Journal of supply chain management* 36.1, pp. 63–69.

Poli, Riccardo, James Kennedy, and Tim Blackwell (2007). "Particle swarm optimization". In: *Swarm intelligence* 1.1, pp. 33–57.

Powell, Michael James David (1977). "Restart procedures for the conjugate gradient method". In: *Mathematical programming* 12.1, pp. 241–254.

Powell, Michael JD (1978). "A fast algorithm for nonlinearly constrained optimization calculations". In: *Numerical analysis*. Springer, pp. 144–157.

Prince, Simon JD and James H Elder (2007). "Probabilistic linear discriminant analysis for inferences about identity". In: *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*. IEEE, pp. 1–8.

Quinlan, J. Ross (1986). "Induction of decision trees". In: *Machine learning* 1.1, pp. 81–106.

— (1987). "Simplifying decision trees". In: *International journal of man-machine studies* 27.3, pp. 221–234.

Raghavan, Balaji et al. (2013). "Towards a space reduction approach for efficient structural shape optimization". In: *Structural and Multidisciplinary Optimization* 48.5, pp. 987–1000.

Rajeev, S and CS Krishnamoorthy (1992). "Discrete optimization of structures using genetic algorithms". In: *Journal of structural engineering* 118.5, pp. 1233–1250.

Reynolds, Robert G (1994). "An introduction to cultural algorithms". In: *Proceedings of the third annual conference on evolutionary programming*. World Scientific, pp. 131–139.

Ros, Raymond and Nikolaus Hansen (2008). "A simple modification in CMA-ES achieving linear time and space complexity". In: *International Conference on Parallel Problem Solving from Nature*. Springer, pp. 296–305.

Ross, Sheldon M (2014). *Introduction to probability models*. Academic press.

Roweis, Sam T and Lawrence K Saul (2000). "Nonlinear dimensionality reduction by locally linear embedding". In: *Science* 290.5500, pp. 2323–2326.

Rozvany, George IN (1996). "Difficulties in truss topology optimization with stress, local buckling and system stability constraints". In: *Structural Optimization* 11.3-4, pp. 213–217.

Rozvany, George IN, Ming Zhou, and Torben Birker (1992). "Generalized shape optimization without homogenization". In: *Structural optimization* 4.3-4, pp. 250–252.

Schalkoff, Robert J (1997). *Artificial neural networks*. Vol. 1. McGraw-Hill New York.

Schölkopf, Bernhard, Alexander Smola, and Klaus-Robert Müller (1998). "Nonlinear component analysis as a kernel eigenvalue problem". In: *Neural computation* 10.5, pp. 1299–1319.

Shakhnarovich, Gregory, Trevor Darrell, and Piotr Indyk (2006). *Nearest-neighbor methods in learning and vision: theory and practice (neural information processing)*. The MIT press.

Shanno, David F (1970). "Conditioning of quasi-Newton methods for function minimization". In: *Mathematics of computation* 24.111, pp. 647–656.

Sigmund, Ole (2001). "A 99 line topology optimization code written in Matlab". In: *Structural and multidisciplinary optimization* 21.2, pp. 120–127.

Sloane, Douglas and S Philip Morgan (1996). "An introduction to categorical data analysis". In: *Annual review of sociology*, pp. 351–375.

Snyman, Jan (2005). *Practical mathematical optimization: an introduction to basic optimization theory and classical and new gradient-based algorithms*. Vol. 97. Springer Science & Business Media.

Sokolowski, Jan and Jean-Paul Zolesio (1992). *Introduction to shape optimization*. Springer.

Srinivas, Nidamarthi and Kalyanmoy Deb (1994). "Muiltiobjective optimization using nondominated sorting in genetic algorithms". In: *Evolutionary computation* 2.3, pp. 221–248.

Stegmann, Jan and Erik Lund (2005). "Discrete material optimization of general composite shell structures". In: *International Journal for Numerical Methods in Engineering* 62.14, pp. 2009–2027.

Storn, Rainer and Kenneth Price (1997). "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces". In: *Journal of global optimization* 11.4, pp. 341–359.

Suykens, Johan AK and Joos Vandewalle (1999). "Least squares support vector machine classifiers". In: *Neural processing letters* 9.3, pp. 293–300.

Svanberg, K (1993). "The method of moving asymptotes (MMA) with some extensions". In: *Optimization of large structural systems*. Springer, pp. 555–566.

Svanberg, Krister (1987). "The method of moving asymptotes—a new method for structural optimization". In: *International journal for numerical methods in engineering* 24.2, pp. 359–373.

— (1995). "A globally convergent version of MMA without linesearch". In: *Proceedings of the first world congress of structural and multidisciplinary optimization*. Vol. 28. Goslar, Germany, pp. 9–16.

Tenenbaum, Joshua B (1998). "Mapping a manifold of perceptual observations". In: *Advances in neural information processing systems*, pp. 682–688.

Tenenbaum, Joshua B, Vin De Silva, and John C Langford (2000). "A global geometric framework for nonlinear dimensionality reduction". In: *science* 290.5500, pp. 2319–2323.

Vapnik, Vladimir (2013). *The nature of statistical learning theory*. Springer science & business media.

Vert, Jean-Philippe, Koji Tsuda, and Bernhard Schölkopf (2004). "A primer on kernel methods". In: *Kernel methods in computational biology* 47, pp. 35–70.

Wall, Wolfgang A, Moritz A Frenzel, and Christian Cyron (2008). "Isogeometric structural shape optimization". In: *Computer methods in applied mechanics and engineering* 197.33-40, pp. 2976–2988.

Wang, Qiong et al. (2007). "Naive Bayesian classifier for rapid assignment of rRNA sequences into the new bacterial taxonomy". In: *Applied and environmental microbiology* 73.16, pp. 5261–5267.

Wang, Ruiping et al. (2008). "Manifold-manifold distance with application to face recognition based on image set". In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, pp. 1–8.

Wang, Xiaoxia, Peter Tiňo, and Mark A Fardal (2008). "Multiple manifolds learning framework based on hierarchical mixture density model". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, pp. 566–581.

Wang, Yong et al. (2011). "Spectral clustering on multiple manifolds". In: *IEEE Transactions on Neural Networks* 22.7, pp. 1149–1161.

Wedderburn, Robert WM (1974). "Quasi-likelihood functions, generalized linear models, and the Gauss—Newton method". In: *Biometrika* 61.3, pp. 439–447.

Wold, Svante, Kim Esbensen, and Paul Geladi (1987). "Principal component analysis". In: *Chemometrics and intelligent laboratory systems* 2.1-3, pp. 37–52.

Wolfe, Philip (1962). *The reduced-gradient method*.

Xie, Yi Min and Grant P Steven (1997). "Basic evolutionary structural optimization". In: *Evolutionary Structural Optimization*. Springer, pp. 12–29.

Yamada, Isao (2001). "The hybrid steepest descent method for the variational inequality problem over the intersection of fixed point sets of nonexpansive mappings". In: *Inherently parallel algorithms in feasibility and optimization and their applications* 8, pp. 473–504.

Yegnanarayana, B (2009). *Artificial neural networks*. PHI Learning Pvt. Ltd.

Yue, H Henry and Masayuki Tomoyasu (2004). "Weighted principal component analysis and its applications to improve FDC performance". In: *Decision and Control, 2004. CDC. 43rd IEEE Conference on*. Vol. 4. IEEE, pp. 4262–4267.

Zhan, Zhi-Hui et al. (2011). "Orthogonal learning particle swarm optimization". In: *IEEE transactions on evolutionary computation* 15.6, pp. 832–847.

Zhang, WH et al. (1996). "A generalized method of moving asymptotes (GMMA) including equality constraints". In: *Structural optimization* 12.2-3, pp. 143–146.

Zhang, Yongyue, Michael Brady, and Stephen Smith (2001). "Segmentation of brain MR images through a hidden Markov random field model and the expectation-maximization algorithm". In: *IEEE transactions on medical imaging* 20.1, pp. 45–57.

Zhao, Xiuyang et al. (2011). "Adaptive knot placement using a GMM-based continuous optimization algorithm in B-spline curve approximation". In: *Computer-Aided Design* 43.6, pp. 598–604.

Zhou, Chang-Chun, Guo-Fu Yin, and Xiao-Bing Hu (2009). "Multi-objective optimization of material selection for sustainable products: artificial neural networks and genetic algorithm approach". In: *Materials & Design* 30.4, pp. 1209–1215.

Zhu, Ciyou et al. (1997). "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization". In: *ACM Transactions on Mathematical Software (TOMS)* 23.4, pp. 550–560.

Zillober, Christian (1993). "A globally convergent version of the method of moving asymptotes". In: *Structural optimization* 6.3, pp. 166–174.

Zoutendijk, Guus (1960). *Methods of feasible directions: a study in linear and non-linear programming*. Elsevier.