Optimisation par essaim de particules: application au clustering des données de grandes dimensions

par

Yanping Lu

Thèse en cotutelle présentée

au Département d'informatique en vue
de l'obtention du grade de Docteur ès sciences (Ph.D.)
FACULTÉ DES SCIENCES, UNIVERSITÉ DE SHERBROOKE

au Département d'informatique en vue
de l'obtention du grade de Docteur ès sciences (Ph.D.)
UNIVERSITÉ DE XIAMEN

5 août 2009

Your file Votre référence
ISBN: 978-0-494-52842-6
Our file Notre référence
ISBN: 978-0-494-52842-6

Sommaire

Le clustering est une tâche difficile de forage de données dans des applications où les données impliquées sont de grandes dimensions. Dans les applications réelles, chaque objet de données est souvent représenté par un vecteur des caractéristiques (ou attributs) dont le nombre peutêtre très élevé. Par exemple, pour représenter un texte on utilise un vecteur de grande taille dont les éléments représentent les fréquences des mots. Les algorithmes traditionnels de clustering ont beaucoup de difficulté quand la dimensionnalité est grande, leurs résultats détériorent rapidement à mesure que le nombre de caractéristiques augmente. Le phénomène s'appelle la malédiction de la dimensionnalité (curse of dimensionality). En effet, quand le nombre de caractéristiques devient grand, les données deviennent très clairsemées et les mesures de distance dans l'espace entier deviennent non significatives. Dans ces cas, certaines caractéristiques peuvent être non pertinentes ou superflues pour certains clusters, et de différents sous-ensembles de caractéristiques peuvent être appropriés pour différents clusters. Ainsi, des clusters se trouvent dans différents sous-espaces de caractéristiques plutôt que dans l'espace de toutes les caractéristiques. Les méthodes visant à trouver des clusters dans différents sous-espaces de caractéristiques sont appelées clustering de sous-espace ou clustering projectif. Cependant, la performance des algorithmes de clustering de sous-espace ou projectif diminue rapidement avec la taille (dimension) des sous-espaces dans lequel les clusters se trouvent. Aussi, beaucoup d'entre eux nécessitent des informations a priori, fournies pars l'usager, pour les aider à déterminer les valeurs de leurs paramètres. Ces informations incluent la distance maximale entre les valeurs d'une dimension, les seuils tels que la densité minimale et la moyenne des dimensions à retenir pour les clusters, etc., qui sont en général difficiles à estimer.

Le but principal de cette thèse est de développer une nouvelle méthode, en se basant sur l'optimisation par Essaim de Particules (PSO pour Particle Swarm Optimization), pour le clustering des données de grandes dimensions. Premièrement, nous avons étudié les principales causes de la convergence prématurée de PSO et proposé une nouvelle version de l'algorithme PSO améliorée que l'on appelle InformPSO. InformPSO est basée sur des principes de diffusion adaptative et de mutation hybride. En s'inspirant de la physique de diffusion d'information, nous avons conçu une fonction pour obtenir une meilleure diversité de particules en tenant compte de leurs distributions et de leur nombre de générations évolutives et en ajustant leurs " habilités cognitives sociales ". En nous basant sur l'auto organisation génétique et l'évolution de chaos, nous avons intégré la sélection clonale dans InformPSO pour implanter l'évolution locale du meilleur candidat particule, *gBest*, et fait usage d'une séquence de logistique pour contrôler la dérive aléatoire du *gBest*. Ces techniques contribuent grandement à éviter des optimums locaux. La convergence globale de l'algorithme est prouvée en utilisant le théorème de chaîne de Markov. Nos expériences sur l'optimisation des fonctions d'étalonnage unimodales et multimodales démontrent que, comparé aux autres variantes de PSO, InformPSO converge plus rapidement et résulte en de meilleurs optimums. Il est plus robuste et plus effectif à empêcher la convergence prématurée.

Par la suite, nous avons étudié deux des principaux problèmes du clustering des données de grandes dimensions, à savoir le problème de pondération de variables dans ce qu'on appelle le " soft " clustering projectif avec un nombre fixé (ou connu) de clusters et le problème même de détermination du nombre de clusters. Nous avons proposé des fonctions objectives spéciales et des schémas de codage adaptés pour permettre d'utiliser le PSO dans la résolution de ces problèmes. Plus précisément, le premier problème, avec le nombre de clusters préfixé et qui vise à trouver un ensemble de poids pour chaque cluster, est formulé comme un problème d'optimisation non linéaire avec des variables continues, sous contraintes de limites. Un nouvel algorithme, appelé PSOVW, est proposé pour chercher les valeurs de poids optimales pour les clusters. Dans PSOVW, nous avons conçu une fonction d'objectif de type k-moyenne impliquant les poids dont les variations sont exponentiellement reflétées. Nous transformons également les contraintes d'égalité initiales en des contraintes de limites en utilisant une représentation non normalisée des poids variables. Nous utilisons ensuite un optimisateur PSO pour minimiser la fonction objective. Nos résultats expérimentaux sur des données synthétiques et des données réelles démontrent que notre algorithme améliore significativement la qualité des clusters trouvés. De plus, les résultats du nouvel algorithme dépendent beaucoup moins des centres initiaux des clusters.

Le deuxième problème vise à déterminer automatiquement le nombre de clusters k et de trouver les clusters en même temps. Ce problème est aussi formulé comme un problème d'optimisation non linéaire avec des contraintes de limites. Pour ce problème de la détermination automatique de k, qui est problématique pour la plupart des algorithmes existants, nous avons proposé un nouvel algorithme de PSO appelé l'autoPSO. Un codage spécial des particules est introduit dans l'autoPSO pour représenter des partitions avec différents nombres de clusters dans la même population. L'index de DB est utilisé comme fonction objective pour mesurer la qualité des partitions avec des nombres semblables ou différents de clusters. L'algorithme autoPSO est testé sur des ensembles de données synthétiques de grandes dimensions et des ensembles de données artificielles de petites dimensions. Ses performances ont été comparées à celles d'autres techniques de clustering. Les résultats expérimentaux indiquent que l'algorithme autoPSO a un potentiel intéressant pour résoudre le problème de clustering des données de grandes dimensions sans le préréglage du nombre de clusters.

Particle Swarm Optimizer: Applications in High-dimensional Data Clustering

By

Yanping Lu

Co-advised thesis submitted

to Département d'informatique
for the degree of Doctor of Science (Ph.D.)
FACULTÉ DES SCIENCES, UNIVERSITÉ DE SHERBROOKE

to Department of Computer Science
for the degree of Doctor of Science (Ph.D.)
XIAMEN UNIVERSITY

August 5, 2009

Abstract

Clustering high-dimensional data is an important but difficult task in various data mining applications. A fundamental starting point for data mining is the assumption that a data object, such as text document, can be represented as a high-dimensional feature vector. Traditional clustering algorithms struggle with high-dimensional data because the quality of results deteriorates due to the curse of dimensionality. As the number of features increases, data becomes very sparse and distance measures in the whole feature space become meaningless. Usually, in a high-dimensional data set, some features may be irrelevant or redundant for clusters and different sets of features may be relevant for different clusters. Thus, clusters can often be found in different feature subsets rather than the whole feature space. Clustering for such data sets is called subspace clustering or projected clustering, aimed at finding clusters from different feature subspaces. On the other hand, the performance of many subspace/projected clustering algorithms drops quickly with the size of the subspaces in which the clusters are found. Also, many of them require domain knowledge provided by the user to help select and tune their settings, like the maximum distance between dimensional values, the threshold of input parameters and the minimum density, which are difficult to set.

Developing effective particle swarm optimization (PSO) for clustering high-dimensional data is the main focus of this thesis. First, in order to improve the performance of the conventional PSO algorithm, we analyze the main causes of the premature convergence and propose a novel PSO algorithm, call InformPSO, based on principles of adaptive diffusion and hybrid mutation. Inspired by the physics of information diffusion, we design a function to achieve a better particle diversity, by taking into account their distribution and the number of evolutionary generations and by adjusting their "social cognitive" abilities. Based on genetic self-organization and chaos evolution, we build clonal selection into InformPSO to implement local evolution of the best particle candidate, gBest, and make use of a Logistic sequence to control the random drift of gBest. These techniques greatly contribute to breaking away from local optima. The global convergence of the algorithm is proved using the theorem of Markov chain. Experiments on optimization of unimodal and multimodal benchmark functions show that, comparing with some other PSO variants, InformPSO converges faster, results in better optima, is more robust, and prevents more effectively the premature convergence.

Then, special treatments of objective functions and encoding schemes are proposed to tailor PSO for two problems commonly encountered in studies related to high-dimensional data clustering. The first problem is the variable weighting problem in soft projected clustering with known the number of clusters k. With presetting the number of clusters k, the problem aims at finding a set of variable weights for each cluster and is formulated as a nonlinear continuous optimization problem subjected to bound constraints. A new algorithm, called PSOVW, is proposed to achieve optimal variable weights for clusters. In PSOVW, we design a suitable k-means objective weighting function, in which a change of variable weights is exponentially reflected. We also transform the original constrained variable weighting problem into a problem with bound constraints, using a non-normalized

representation of variable weights, and we utilize a particle swarm optimizer to minimize the objective function in order to obtain global optima to the variable weighting problem in clustering. Our experimental results on both synthetic and real data show that the proposed algorithm greatly improves cluster quality. In addition, the results of the new algorithm are much less dependent on the initial cluster centroids.

The latter problem aims at automatically determining the number of clusters k as well as identifying clusters. Also, it is formulated as a nonlinear optimization problem with bound constraints. For the problem of automatical determination of k, which is troublesome to most clustering algorithms, a PSO algorithm called autoPSO is proposed. A special coding of particles is introduced into autoPSO to represent partitions with different numbers of clusters in the same population. The DB index is employed as the objective function to measure the quality of partitions with similar or different numbers of clusters. autoPSO is carried out on both synthetic high-dimensional datasets and handcrafted low-dimensional datasets and its performance is compared to other selected clustering techniques. Experimental results indicate that the promising potential pertaining to autoPSO applicability to clustering high-dimensional data without the preset number of clusters k.

Le 5 août 2009

le jury a accepté la thèse de Mme Yanping Lu dans sa version finale.

Membres du jury

M. Shengrui Wang
Directeur
Département d'informatique

M. Changle Zhou
Codirecteur
-Xiamen University

M. Shaozi Li
Membre
Xiamen University

M. Ernest Monga
Membre
Département de mathématiques

M. Guolong Chen
Membre externe
Fuzhou University

M. Jean-Pierre Dussault
Président-rapporteur
Département d'informatique

Acknowledgements

First of all, I would like to thank my supervisors Prof. Shengrui Wang, Prof. Shaozi Li and Prof. Changle Zhou for their valuable support, help and feedback throughout the progress of this work.

Thanks to everyone in the Laboratory for Intelligence Multimedia for making the first two years of my PhD life enjoyable in Xiamen University, particularly Songzhi Su, Li Wang, Zhengming Xu, Zhen Da and thanks to Tengke Xiong and Shu Wu who have offered their help on numerous occasions in University of Sherbrooke.

Finally, I am very much obliged to China Scholarship Council (CSC), Prof. Shengrui Wang and Prof. Shaozi Li, who have supported me with doctoral scholarships.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Clustering high-dimensional data is a common but important task in various data mining applications. A fundamental starting point for data mining is the assumption that a data object can be represented as a high-dimensional feature vector. Text clustering is a typical example. In text mining, a text data set is viewed as a matrix, in which a row represents a document and a column represents a unique term. The number of dimensions corresponds to the number of unique terms, which is usually in the hundreds or thousands. Another application where high-dimensional data occurs is insurance company customer prediction. It is important to separate potential customers into groups to help companies predict who would be interested in buying an insurance policy (62). Many other applications such as bankruptcy prediction, web mining, protein function prediction, etc., present similar data analysis problems.

Clustering high-dimensional data is a difficult task because clusters of high-dimensional data are usually embedded in lower-dimensional subspaces and feature subspaces for different clusters can overlap. In a text data set, documents related to a particular topic are chacterized by one subset of terms. For example, a group of documents are categorized under the topic, *electronics*, because they contain a subset of terms such as *electronics*, *signal*, *circuit*, etc. The terms describing another topic, *athlete*, may not occur in the documents on *electronics* but will occur in the documents related to *sports*.

Traditional clustering algorithms struggle with high-dimensional data because the quality of results deteriorates due to the curse of dimensionality. As the number of dimensions increases, data becomes very sparse and distance measures in the whole dimension space become meaningless. Irrelevant dimensions spread out the data points until they are almost equidistant from each other in very high dimensions. The phenomenon is exacerbated when objects are related in different ways in different feature subsets. In fact, some dimensions may be irrelevant or redundant for certain clusters, and different sets of dimensions may be relevant for different clusters. Thus, clusters should often be searched for in subspaces of dimensions rather than the whole dimension space.

Clustering of such data sets uses an approach called subspace clustering or projected clustering, aimed at finding clusters from different subspaces. Subspace clustering in general seeks to identify all the subspaces of the dimension space where clusters are most well-separated (see for instance (35; 51; 54)). The terms subspace clustering and projected clustering are not always used in a consistent way in the literature, but as a general rule, subspace clustering algorithms compute overlapping clusters, whereas projected clustering aims to partition the data set into disjoint clusters (see for instance (12; 18; 22; 23)). Often, projected clustering algorithms search for clusters in subspaces, each of which is spanned by a number of base vectors (main axes). The performance of many subspace/projected clustering algorithms drops quickly with the size of the subspaces in which the clusters are found (37). Also, many of them require domain knowledge provided by the user to help select and tune their settings, such as the maximum distance between dimensional values (12), the thresholds of input parameters (22; 23) and the minimum density (51; 54), which are difficult to establish.

Recently, a number of soft projected clustering algorithms have been developed to identify clusters by assigning an optimal variable weight vector to each cluster (14; 15; 32; 36). Each of these algorithms iteratively minimizes an objective function. Although the cluster membership of an object is calculated in the whole variable space, the similarity between each pair of objects is based on weighted variable differences. The variable weights transform distance so that the associated cluster is reshaped into a dense hypersphere and can be separated from other clusters. Soft projected clustering algorithms are driven by evaluation criteria and search strategies. Consequently, defining the objective function and efficiently determining the optimal variable weights are the two most important issues in soft projected clustering.

Another fundamental difficulty in clustering high-dimensional data concerns how an algorithm automatically determines the number of clusters k in the data set. Most existing subspace algorithms require the number of clusters k as an input parameter, and this is usually very difficult to set where the structure of the data set is completely unknown. While a number of different clustering approaches to automatically determine the number of clusters have been proposed (7; 8; 31; 39; 47), no reliable method exists for clustering high-dimensional data.

1.2 Thesis Objectives

Developing effective algorithms for clustering high-dimensional data is the main focus of this thesis. The goal is to address two main problems, namely the variable weighting problem in soft projected clustering with a number of clusters k, and the problem of automatic determination of k. Given a preset number of clusters k, the first of these is formulated as a nonlinear continuous optimization problem subjected to bound constraints. The aim is to find a set of variable weights for each cluster. The second problem is also formulated as a non-linear constrained continuous optimization problem. The aim is to find a set of cluster centers with the correct number of clusters. The objective criteria, the encoding of particles and searching techniques are addressed. The inherent properties of high-dimensional data, such as sparsity and equidistance, make cluster identification a rather tedious task. This emphasizes the need to develop modern tools to analyze and design algorithms for clustering high-dimensional data. In this thesis, the goal is to develop particle swarm optimization techniques as new heuristic methods with global search capabilities, to solve the problem of cluster identification. An attempt is made to modify the original PSO algorithm, in order to explore its potential suitability for some of the optimization problems that exist in the area of clustering. Finally, the performances of the improved PSO algorithms is investigated and compared to that of other algorithms commonly used for clustering high-dimensional data.

1.3 Organization of the Thesis

The main tasks involved in the development of this thesis are detailed in five chapters. Chapter 1 highlights the motivation behind the thesis and introduces its main objectives. The concepts of particle swarm optimization theory are described in Chapter 2, which presents the development of the theory and its advantages over other global optimization approaches. In addition, an improved particle swarm optimizer to overcome the premature convergence problem is described in this chapter, and experimental results on benchmark tests for comparison of other PSO variants are also presented. Chapter 3 includes a recent and detailed state of the art review on the variable weighting problem in soft subspace clustering, proposes an advanced PSO algorithm for the variable weighting problem and, finally, presents experiments comparing the approach with other soft projected clustering algorithms on both synthetic and real-life data sets. The application of the new algorithm to text clustering is described in detail in this chapter. Chapter 4 reviews the literature on recently proposed approaches to solve the problem of determining of the number of clusters k and addresses automatic determination of k by another enhanced PSO algorithm. The enhanced algorithm is compared to other techniques and its effectiveness on synthetic and real-life data sets is assessed. Chapter 5 gives concluding remarks and presents the main contributions of the thesis and possible future extensions to the thesis research.

Chapter 2

Particle Swarm Optimization

2.1 Optimization and Swarm Intelligence

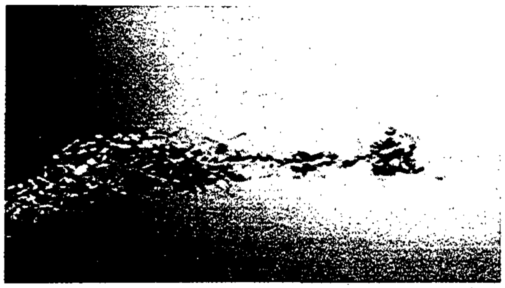
Optimization has been an active area of research for several decades. Although there have been many local optimization methods, better global optimization algorithms are always needed because real-world optimization problems are becoming increasingly complex. Optimization problems with bound constraints can be formulated as a d-dimensional minimization problem via the following formula:

$$\min F(x), x = [x^1, x^2, ..., x^d], u \leq x \leq l \tag{2.1}$$

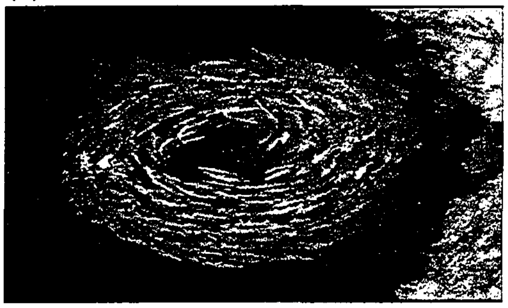
where d is the number of parameters to be optimized, u and l represent the lower and upper bound of the search space, respectively.

Swarm intelligence (SI) is one of the advanced global heuristic techniques used for complex optimization problems. SI is artificial intelligence based on the collective behavior of decentralized, self-organized systems. The expression was introduced by Gerardo Beni and Jing Wang in 1989, in the context of cellular robotic systems.

SI systems are typically made up of a population of simple agents interacting locally with one another and with their environment. The agents follow very simple rules, and although there is no centralized control structure dictating how individual agents should behave, local interactions between such agents lead to the emergence of complex global behavior. Natural examples of SI include ant colony behavior, bird flocking, animal herding, bacterial growth, and fish schooling (see Figure 2.1).


(a) ant chains

(b) ant wall


(c) fish schooling

(d) birds flocking

Figure 2.1: Several models of collective behavior: (a) ant chains, (b) ant wall, (c) fish schooling, (d) birds flocking.

Since 1990, several algorithms inspired by collective behavior (such as that of social insects, or bird flocking) have been proposed. The application areas of these algorithms include such well-studied optimization problems as NP-hard problems (the Traveling Salesman Problem, the Quadratic Assignment Problem, graph problems), network routing, clustering, data mining, job scheduling, etc. In this thesis, we are most interested in Particle Swarm Optimization (PSO), which is currently one of the most popular algorithms in the swarm intelligence domain.

2.2 PSO and its Developments

2.2.1 Particle Swarm Optimization (PSO)

PSO is a population-based stochastic optimization technique developed by Eberhart and Kennedy in 1995 (16), inspired by the social behavior of insects, animal herding, bird flocking and fish schooling, where these swarms search for food in a collaborative manner. PSO shares many similarities with evolutionary computation techniques such as genetic algorithms (GA). The system is initialized with a population of random solutions and searches for optima by successive updating. However, unlike GA, the potential solutions are called particles in PSO and each particle is also associated with a position and a velocity. In PSO, there are no genetic operators such as crossover and mutation. Particles fly through the problem space with velocities which are dynamically adjusted according to the current optimal particles.

In PSO, an individual in the swarm, called a particle, represents a potential solution which is usually a point in the search space. Each particle adapts its search patterns by learning from its own experience and that of other particles. These phenomena are studied and their mathematical models are constructed. Each particle has a fitness value and a velocity to adjust its flying direction and the particles fly through the problem space by learning from the best experiences of all the particles. The particles thus have a tendency to fly towards a better search area and search for the global optimum in the d-dimensional solution space.

The velocity and position updates of the i^{th} particle are as follows:

$$V_i(t+1) = w * V_i(t) + c1 * r1 * (pBest_i - X_i(t)) + c2 * r2 * (gBest - X_i(t)) \quad (2.2)$$

$$X_i(t+1) = V_i(t+1) + X_i(t) \quad (2.3)$$

where X_i is the position of the i^{th} particle, V_i represents its velocity and $pBest_i$ is the best previous position, i.e. the one that yields the best fitness value for the particle. $gBest$ is the best position discovered by the whole population and

w is the inertia weight used to balance between global and local search abilities. Basically, a high inertia weight is more appropriate for global search, and a low inertia weight facilitates local search. $c1$ and $c2$ are the acceleration constants, which represent the weighting of stochastic acceleration terms that pull each particle towards the *pBest* and *gBest* positions. $r1$ and $r2$ are two random numbers in the range [0, 1]. A particle's velocity on each dimension is restricted to a maximum magnitude. If V_i exceeds a positive constant value V_{max} specified by the user, then the velocity is assigned to V_{max}.

2.2.2 PSO Developments

The PSO algorithm is simple in concept, easy to implement and computationally efficient. Since its introduction in 1995 by Kennedy and Eberhart (16), PSO has generated considerable interest and has been empirically demonstrated to perform well on many optimization problems. However, it may easily get trapped in local optima when solving non-linear optimization problems. Research into improving the performance of PSO continues to receive a lot of attention. In order to enhance the performance of PSO on non-linear optimization problems, many researchers have made modifications to the original algorithm deriving many interesting variants..

Some of these improve the performance of the original PSO by introducing different parameters into its velocity update. The original algorithm includes a parameter called inertia weight Formula2.2 whose function is to balance the global and local search abilities. A high inertia weight is more appropriate for global search, and a low inertia weight facilitates local search. Shi and Eberhart (65) in 1998 proposed a linearly decreasing inertia weight over the course of search. Fuzzy methods for modifying the inertia weight were designed by Shi and Eberhart (66). In addition to the time-varying inertia weight, Fan in (20) introduced a linearly decreasing scheme. Another parameter, called the constriction factor, was introduced by Clerc in (41), by analyzing the convergence behavior of the

PSO. The constriction factor guarantees convergence and makes the algorithm converge rapidly.

Improving PSOs performance by designing different types of topologies has been an active research direction. Kennedy (33) claimed that PSO with a small neighborhood might perform better on complex problems, while PSO with a large neighborhood would perform better on simple problems. Suganthan (56) applied a dynamic adjustment whereby the neighborhood of a particle gradually increases until it includes all particles. In (29), Hu and Eberhart also used a dynamic neighborhood where the closest particles in the performance space are selected to be the new neighborhood in each generation. Parsopoulos and Vrahatis combined the global and local versions together to construct a unified particle swarm optimizer (UPSO) (49). Mendes and Kennedy introduced a fully informed PSO in (52). Instead of using the best previous position *gBest* of the swarm in the standard algorithm, all the neighbors of the particle are used to update the velocity. The influence of each particle on its neighbors is weighted based on its fitness value and the neighborhood size. Veeramachaneni et al. developed the fitness-distance-ratio-based PSO (FDR-PSO) with near neighbor interactions (58). When updating each velocity dimension, the FDR-PSO algorithm selects another particle which has a higher fitness value and is closer to the particle being updated.

Some researchers have investigated hybridization by combining PSO with other search techniques to improve its performance. Evolutionary operators such as selection, crossover, and mutation have been introduced into PSO to keep the best particles (6), increase the diversity of the population (30), and improve the ability to escape local minima (42). Mutation operators are also used to mutate parameters such as the inertia weight (46). Other approaches that have been investigated include relocating the particles when they are too close to each other (40) or using collision-avoiding mechanisms (10) to prevent particles from moving too close to each other in order to maintain diversity and escape from local optima. In (42), the swarm is divided into subpopulations, and a breeding operator is used within a subpopulation or between the subpopulations to increase the diversity of the population. Negative entropy is used to discourage premature convergence in (63). In (49), deflection, stretching, and repulsion techniques are used to find as many minima as possible by preventing particles from moving to a previously discovered minimal region. Recently, a cooperative particle swarm optimizer (CPSO-H) (60) was proposed. Although CPSO-H uses one-dimensional (1-D) swarms to search each dimension separately, the results of these searches are integrated by a global swarm to significantly improve the performance of the original PSO on multimodal problems.

The original PSO and its variants can generate a high-quality solution within an acceptable calculation time and with a stable convergence characteristic in many problems where most other methods fail to converge. It can thus be effectively applied to various optimization problems. A number of papers have been published in the past few years that focus on the applications of PSO, such as neural network training (61), power and voltage control (27), task assignment (5), single machine total weighted tardiness problems (45) and automated drilling (24).

2.3 InformPSO

2.3.1 Motivation

Although there are numerous versions of PSO which improve the performance of the original PSO to a certain degree, premature convergence when solving complex optimization problems is still the main deficiency of PSO. In the original PSO algorithm, each particle learns from its *pBest* and *gBest* to obtain the personal and social cognitions simultaneously and then changes its velocity. However, applying the same social cognition aspect to all particles in the swarm only makes the original PSO converge fast and appears to be a somewhat arbitrary decision. Each particle obtains the same information from the *gBest* as others even if it is far from the *gBest*. In such situations, particles may be quickly attracted by the *gBest*, resulting in a rapid loss of population diversity.Consequently, the algorithm easily becomes trapped in a local optimum if the *gBest* is in a complex search environment with numerous local optima.

Another cause of the premature convergence problem plaguing the original PSO is that the *pBest* and *gBest* do not contribute to *gBest* in the velocity update equation. *gBest*, the best particle in the original PSO, always flies in the direction of its previous velocity, which means it easily become trapped in a local optimum and is unable to break away from it. In other words, *gBest*, as the most important memory unit, is restricted to learning from particles with better fitness values and does not employ the self-learning mechanism.

In order to increase the population diversity of the original PSO and improve the ability to get away from local optima, we have to propose a new variant to discourage premature convergence and improve on the performance of the original version. In our algorithm, an information diffusion function is introduced into the social cognition part of the velocity update. The function ensures that the diversity of the swarm is preserved to discourage premature convergence. Another mutation strategy is employed to improve *gBest*'s ability to break away from local optima.

2.3.2 InformPSO

Information diffusion function

In fact, information diffusion among biological particles is a time-dependent process. Obviously, particles close to the current best particle *gBest* quickly change their direction and rate of velocities towards it, while other particles far from *gBest* move more slowly towards it or break away from its effect. On the assumption that information is diffused among particles in a short time, information received by particles close to *gBest* is more than that received by those far from it. Therefore, an information diffusion function, related to degree of membership with respect to the "surroundings" of $gBest$, is incorporated into the velocity update in the original PSO to adjust the variable "social cognition" aspect of particles.

In this improved version of PSO, the velocity update is expressed as follows:

$$V_i(t+1) = w \cdot V_i(t) + c1 \cdot r1 \cdot (pBest_i - X_i) + H_i \cdot c2 \cdot r2 \cdot (gBest - X_i) \quad (2.4)$$

$$H_i = [1 - \frac{d_i + 1}{\max_{1 \leq j \leq n} d_j + 1}] \cdot \frac{n+1}{n+n'+1} \cdot (1 - \frac{t}{T}) \quad (2.5)$$

where H_i is an information diffusion function, d_i is the distance between particle i and $gBest$, t represents the current number of evolution generations, and T is the total number of generations and the overall number of generations specified by the user, respectively.

The scale of motion particles towards $gBest$ in the information diffusion function consists of three sections, namely,

- $1 - \frac{d_i+1}{\max_{1 \leq j \leq n} d_j + 1}$ combines the scale of motion with the membership degree of with respect to the "surroundings" of $gBest$. Particles close to $gBest$ move quickly towards it, while particles far from it move slowly towards it. As a result, particles in the same generation move towards $gBest$ according to variable scales to maintain the population diversity. d_i is the distance between i^{th} particle and $gBest$ and here the distance is measured by their position difference. "1" is added to prevent the denominator from being zero.

- $\frac{n+1}{n+n'+1}$ combines the scale of motion with the particle distribution. During the search procedure, the distribution of particles is examined. Once the number of particles very close to $gBest$ exceeds a threshold value specified by the user, the scale of motion becomes small to avoid losing the population diversity. Here, n' is the user-defined number of particles very close to *gBest*. n is the number of other particles. "1" is added to prevent the function value from being zero.

- $1-\frac{t}{T}$ combines the scale of motion with the number of evolution generations. The earlier particles have lower quality and hence the scale of motion is relatively large, favoring global search by the particles. As the number of generations increases, the scale becomes smaller, favoring local search by the particles and maintaining the population diversity.

By inspecting the equations in 2.4 and 2.5, we understand that particles perform variable-wise velocity update to *gBest*. This operation improves the local search ability of PSO, increases the particles' diversity and enables the swarm to overcome the premature convergence problem.

Clonal selection operator

In non-linear optimization problems, the best position candidate, *gBest*, is often a local optimum, which may give other particles wrong information and lead to premature convergence. Besides learning from particles with better fitness values, *gBest* also needs to be able to move out of local optima, i.e., it requires mutation ability to implement local self-evolution. PSO variants endowed with the deterministic mutation operation can be easily trapped in local optima, while PSO variants incorporating the random mutation operation are unable to implement precise local search. The PSO variants employed with the deterministic mutation operation can be easily trapped in local optima. Based on genetic self-organization and chaos evolution, we build clonal selection into our PSO algorithm to implement local evolution of *gBest* and make use of a logistic sequence to control the random drift of *gBest*. These mutation techniques are simultaneously deterministic and random and hence strongly promote the ability to break away from local optima.

The genetic evolution principle of clonal selection theory can be stated as follows: A gene, as a memory unit which carries genetic information to the next generation, will replicate and grow by self-replicating; but due to the changing environment, copying errors occur frequently to the genetic material during the process of self-replication, leading to genetic mutation. The majority of these mutant genes will have no effect and will die because of the mutation, but one might survive to change the genetic information and thus produce its own offspring. Self-replication of genes in a small region, genetic mutation and mutation death cause the original gene to spread far and wide. Over time the number of genes with this mutation may form a larger percentage of the population. Inspired by the genetic evolution principle, our new algorithm applies the clonal selection operation on the best position *gBest* in the swarm, and thus pulls *gBest* in another direction if it is trapped in a local optimum. The clonal selection operation proceeds as follows:

- The best position in the population, *gBest*, self-replicates into a sub-swarm, in which the individual has the same characteristics as *gBest*.
- Based on a Cauchy distribution, this sub-swarm is edited and mutated in a local region to form a cluster, in which individuals are different and might have different fitness values.
- The individual with the best fitness value is chosen from the new sub-swarm as the *gBest* for the velocity update of the next generation.

From the above description, we can see that the essence of the clonal approach is the evolution of one generation, and near the candidate solution it will create a cluster of mutation solutions depending on the fitness. As a result, *gBest* is improved in a local region by the clonal selection operation, which enables the PSO effectively to break away from local optima. It is also able to more correctly guide other particles to move and convergence is greatly accelerated.

Logistic sequence mutation

Actually, the clonal selection operation is a kind of mutation with small scale. For optimization functions in which local optima are close to the global optima, it can effectively help *gBest* to move out of local optima. However, for functions where local optima are far from the global optima, it only accelerates the algorithm's convergence to some local optimum. So, when *gBest* is trapped in a local optimum and there are no better local optima close by, a random but regular mutation drift must be used to replace the clonal selection operation in order to help *gBest* escape.

The nonlinear time sequence, which has a series of properties such as randomicity and, ergodicity, has been introduced into the evolutionary computation to construct new intelligent algorithms. Once the clonal selection operation ceases improve *gBest* in a local region, the new algorithm makes use of a logistic sequence to control the random drift of *gBest*. What this algorithm needs is a large random drift, i.e. the ergodicity space of the logistic reflection needs to be large. We therefore set two parameters in the logistic sequence, the control parameter r and the initial value $x(0)$, by observing their influence on the space of the logistic inflection ergodicity scope.

A logistic sequence is a nonlinear system:

$$x(t+1) = r \cdot x(t) \cdot [1 - x(t)], r \in R, x(0) \in [0, 1] \tag{2.6}$$

where r is a control parameter. After defining r, we start with an initial arbitrary value and produce a time sequence $x(1), x(2), \cdots$. Formula 2.6 is a deterministic system with no stochastic disturbance.

$$gBest(t+1) = gBest(t) + [r3 < P_m] \cdot x(t) \cdot r4 + [r3 \geq P_m] \cdot x(t) \cdot r5 \tag{2.7}$$

$$[r3 < P_m] = \{ \begin{array}{ll} 1 & \text{if } r3 < P_m \\ 0 & \text{else} \end{array} \tag{2.8}$$

where $r3$, $r4$ and $r5$ are random numbers in the range of [0, 1]. P_m is the mutation probability in the range of [0.05, 0.5]. We experimentally set $P_m = 0.05 + 0.45 \cdot \frac{e^{1:m} - e}{e^m - e}$, where m is the dimension of problem.

Graph (a) of Figure 2.2 shows the chaos process of the logistic sequence with a time increase. When $3 \leq r \leq 4$, the sequence becomes very complicated. When

$r = 3.56$, it will be led to the chaos through a multi-period process. Graphs (b) and (c) of Figure 2.2 describe the influence of the chaos factor r on the space of the logistic inflection ergodicity scope. The solid lines are the sequence's trajectory when $r = 4$, while the broken lines are the sequences trajectory when $r = 3.6$ and $r = 3.9$. Obviously, the former ergodicity scope is wider than the latter ones. Therefore, we set $r = 4$ in the new algorithm. In graph (d), we examine the influence of initial value $x(0)$ on the logistic sequence when $r = 4$. The sequence tends to be zero when $r = 4$ and $x(0) = 0.5$, otherwise it tends towards chaos. Therefore, we take $r = 4$ and a value of $x(0)$ in the range of [0,1], except for 0, 0.5 and 1.

Convergence Analysis

The improved PSO algorithm based on clonal selection and adaptive mutation, termed InformPSO, operates synchronously on two swarms, the particle swarm and the memory swarm (*pBest* and *gBest*). The evolutionary processes of the two swarms run side by side. The particle swarm is the foundation of the search in the search space, so more attention is paid to global search. The information diffusion function adjusts the flying speed of each particle to the best position, *gbest*, and carries out variable local search, in order to achieve a better particle diversity. The best particle candidate, *gBest*, as one of the most important memory units is replaced by better particles to accelerate convergence. On the other hand, local evolution is implemented by the clonal selection operation and a logistic sequence is incorporated to control the random drift of *gBest*. These techniques greatly enhance the ability break away from local optima. The cooperation among these operations makes the new algorithm converge to global optima. We prove the global convergence of InformPSO as follows:

Assume that (1) the problem domain Ω is a bound region in m-dimensional Euclidean space, and (2) the objective function f is continuous on definition domain Ω. This implies the existence of the set of global optima A^* of the function f in its definition domain Ω.

Definition 1. Define the set of global optimal solutions to the problem to be optimized as:

$$A^* = \{A \in S | f(A) \doteq f^* = \min f(A'), A' \in S\}$$

where S is called the population space. For the population A, $\delta(A) = |A \cap A^*|$ gives the number of global optimal solutions in A. $S_1 = \{A \in S_1 | \delta(A) \geq 1\}$ is called the optimal population space, within which there exists at least one optimal individual. $S_0 \Rightarrow S \backslash S_1$ is called the normal population space.

Definition 2. For random state A_0, the algorithm converges to the global optimum with probability 1, if and only if $\lim_{t \to \infty} P\{\delta(A(t)) \geq 1 | A(0) = A_0\} = 1$; or any random sequence $\{\xi_n\}$converges to random variable ξ with probability 1, if $P\{\bigcap_{n=1}^{\infty} \bigcup_{k \geq n} |\xi_k - \xi| \geq r\} = 0$ for $\forall r \geq 0$.

Theorem 1. For the optimization problem with bound constraints $\min f(x)$ $x = (x_1, x_2, \cdots, x_n)$, $s.t. a_i \leq x_i \leq b_i, i = 1, 2, \cdots, n$, the original particle swarm optimization algorithm (PSO) converges to local optima and converges to global optima with probability 1.

Please see references (21; 26).

Theorem 2. For the optimization problem with bound constraints $\min f(x)$, $x = (x_1, x_2, \cdots, x_n)$, $s.t. a_i \leq x_i \leq b_i, i = 1, 2, \cdots, n$, the particle swarm optimization algorithm (InformPSO) converges to local optima and converges to global optima with probability 1.

Proof. InformPSO converges to local optima, see (26).

There exists the set of global optima of the optimization problem with bound constraints. From the main idea of our algorithm, it is based on the original PSO with three operations added to it.

- **Adaptive diffusion operation**: From formula 2.4, the change in positions between two adjacent generations is $\Delta x_i^j = w \cdot v_i + c1 \cdot (pBest_i^j - x_i^j) \cdot r1 + H_i \cdot c1 \cdot (gBest_i^j - x_i^j) \cdot r2$. $r1, r2$ $N(0, \sigma_j)$, so Δ_i^j $N(0, \sigma_j)$.

- **Clonal selection operation**: From the property of the clonal selection operation, where the optimal population won't evolve into the normal one, the following equation holds:

$$P\{\delta(A(t+1)) = 0|\delta(A(t)) \neq 0\} = P_{10} = 0$$

- **Logistic sequence mutation**: Note that $B(x^*, r) = \{x \in \Omega||f(x) - f^*| < r\}, \forall r > 0$ is the neighborhood of the global optimum x^*. Note that $A_t = \{(A(t) \cap B(x^*, r)) \neq \emptyset\}$ represents a random sequence entering $B(x^*, r)$ at the t^{th} iteration. For a given r, the occurrence of A_1 leads to the occurrence of A_2 due to the ergodicity of the logistic sequence. Hence, we have

$$A_1 \subseteq A_2 \subseteq \cdots \subseteq A_t \subseteq \cdots$$

Therefore,

$$P(A_1) \leq P(A_2) \leq \cdots \leq P(A_t) \leq \cdots$$

Note that $0 \leq P(A_t) \leq 1$.

So, $\lim_{t\to\infty} P(A_t)$ exists.

Assume that random sequence $\delta_t = \{ \begin{array}{ll} 1 & \text{entering } B(x^*, r) \text{ at } t^{th} \text{ iteration} \\ 0 & \text{not entering } B(x^*, r) \text{ at } t^{th} \text{ iteration} \end{array}$, $t = 1, 2, \cdots$.

Then $A_t = \{\delta_t = 1\}$.

Let $P\{A_t\} = P\{\delta_t = 1\} = p_t$, $P\{\delta_t = 0\} = 1 - p_t$, $S_t = \frac{1}{t} \cdot \sum_1^t \delta_t$, $t = 1, 2, \cdots$,

Therefore,

$$E(S_t) = \frac{1}{t} \cdot \sum_1^t p_t, D(S_t) = \frac{1}{t^2} \cdot \sum_1^t D(\delta_t) = \frac{1}{t^2} \cdot \sum_1^t p_t \cdot (1 - p_t) \leq \frac{1}{4 \cdot t},$$

where $E(S_t)$ and $D(S_t)$ are the expectation and the variance of sequence S_t, respectively.

From the Chebysher's Inequality,

$$P\{|S_t - E(S_t)| \geq r\} \leq \frac{D^2(S_t)}{r^2} = \frac{1}{4 \cdot m \cdot r^2}$$

.

Consequently,

$$\lim_{t\to\infty} P\{|S_t - E(S_t)| \geq r\} = 0$$

Note that

$$\delta_t = t \cdot S_t - (t-1) \cdot S_{t-1}$$

Thus,

$$\lim_{t\to\infty} P\{|\delta_t - E(\delta_t)| \geq r\} = 0$$

Therefore,

$$P\{\bigcap_{t=1}^{\infty}\bigcup_{k\geq t} |\delta_k - E(\delta_k)| \geq r\} = P\{\lim_{t\to\infty}\bigcup_{k\geq t} |\delta_k - E(\delta_k)| \geq r\} = \lim_{t\to\infty} P\{|\delta_t - E(\delta_t)| \geq r\} = 0$$

So, random sequence $\{\delta_t\}$ converges to a global optimum with probability 1. Also sequence A_t converges to a global optimum with probability 1, namely $P\{\bigcap_{t=1}^{\infty}\bigcup_{k\geq t} A_t\} = 0$. Thus, InformPSO converges to a global optimum with probability 1.

2.4 Validation of Effectiveness

2.4.1 Benchmark Functions

A comprehensive performance study was conducted to evaluate the performance of our PSO variant (InformPSO), in terms of convergence speed and global search capability, on 10 benchmark functions with different properties. Other PSO variants were chosen for comparison. The properties and the formulas of these functions are presented below.

The benchmark functions in Table 2.1 have different characteristics. Functions f_1 to f_5 are continuous, unimodal functions, which are usually used to test the convergence speed of algorithms. The function f_5 Rosenbrockis a typical,

Name	Function	x^*	$f(x^*)$
Sphere	$f_1 = \sum_{i=1}^{D} x_i^2$	[0,0,0]	0
Hyper-ellipsoid	$f_2 = \sum_{i=1}^{D} i \cdot x_i^2$	[0,0,0]	0
Sum of different powers	$f_3 = \sum_{i=1}^{D} \lvert x_i \rvert^{i+1}$	[1,1,1]	0
Rotated hyper-ellipsoid	$f_4 = \sum_{i=1}^{D} [\sum_{j=1}^{i} x_j]^2$	[0,0,0]	0
Rosenbrock	$f_5 = \sum_{i=1}^{D} 100 \cdot (x_{i+1} - x_i^2)^2 + (x_i - 1)^2$	[1,1,1]	0
Griewank	$f_6 = \frac{1}{4000} \cdot \sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{n} \cos \frac{x_i}{\sqrt{i}} + 1$	[0,0,0]	0
Ackley	$f_7 = 20 + e - 20 \cdot e^{-0.2 \cdot \sqrt{\frac{1}{D} \cdot \sum_{i=1}^{D} x_i^2}}$ $-e^{\frac{1}{D} \sum_{i=1}^{D} \cos 2\pi x_i}$	[0,0,0]	0
Rastrigin	$f_8 = \sum_{i=1}^{D} [x_i^2 - 10 \cdot \cos 2\pi x_i + 10]$	[0,0,0]	0
Weierstrass	$f_9 = \sum_{i=1}^{D} \sum_{k=0}^{k_{max}} [a^k \cdot \cos 2\pi b^k (x_i + 0.5)]$ $-D \sum_{k=0}^{k_{max}} [a^k \cdot \cos 2\pi b^k \cdot 0.5]$ $a = 0.5, b = 3, k_{max} = 20$	[0,0,0]	0
Schwefel	$f_{10} = 418.9829 \cdot D - \sum_{i=1}^{D} x_i \sin \sqrt{\lvert x_i \rvert}$	$[420.96, 420.96, 420.96]$	0

Table 2.1: the ten benchmark functions. Functions f_1 to f_5 are unimodal functions, while functions f_6 to f_{10} are multimodal ones [2]. $f(x^*)$ means the global minimum of functions and x^* is the global minimal optimum.

complex optimization problem, whose global minimum is inside a long, narrow, parabolic-shaped flat valley. Finding the valley is trivial, however, converging to the global minimum is difficult. Functions f_6 to f_{10} are complex, nonlinear multimodal problems with a large number of local optima. When attempting to solve functions f_6 to f_{10}, algorithms may easily fall into a local optimum. Hence, an algorithm capable of maintaining a larger population diversity, breaking away from local minima, and avoiding premature convergence is likely to yield better results.

2.4.2 Parameter Setting

To allow fair comparison of InformPSO with other PSO variants, we set the same parameters as in (30; 52). The machine precision is set at 10^{-40} and w decreases with increasing generations from 0.9 to 0.4. $c1$ and $c2$ are both set at 2. Particles' initial positions are restricted by the search range and their velocities are restricted by V_{max}, which is equal to the search range. For other parameter settings, please see Table 2.2. Aside from these common parameters used in PSO variants, there is an additional parameter in InformPSO that needs to be specified. It is the sub-swarm's population size, m. Given that it is unknown whether the function to be optimized is unimodal or multimodal, m is set at 10 for all functions. Throughout this thesis, we run each algorithm implemented by us using MATLAB or VC++ on the following system: Microsoft Windows XP Pro., Intel Core 2 Duo CPU 2.66GHz, 1.96GB of RAM. MATLAB constructs the double data type according to IEEE Standard 754 for double precision. The range of double-precision positive real numbers in MATLAB is $2.22507\times^{-308}$ through $1.79769\times^{308}$.

2.4.3 Experimental Results

In this section, we examined the performances of eight PSO variants including InformPSO on the above ten benchmark functions. We implemented PSO_w, FDR_PSO and InformPSO. Also, we got the MATLAB code of CLPSO from Liang (30). Other interesting variations of the PSO algorithm (described below) have recently been proposed by researchers. Although we have not implemented all of these algorithms, we conducted comparison with them using the results reported in the publications cited below:

- Original PSO (16)
- Modified PSO with inertia weight (PSO_w)(65)
- Local version of PSO with constriction factors (PSO_cf_local) (34)

Function	$InitialRange$	Max_Gen	$Size$	Dim
f_1	[-100, 100]	3000	10	20
f_2	[-5.12, 5.12]	3000	10	10
f_3	[-1, 1]	3000	10	10
f_4	[-65, 65]	3000	10	10
f_5	[-2.048, 2.048]	3000	10	10
f_6	[-600, 600]	3000	10	10
f_7	[-32.768, 32.768]	3000	10	10
f_8	[-5.12, 5.12]	3000	10	10
f_9	[-0.5, 0.5]	3000	10	10
f_{10}	[-500, 500]	3000	10	10

Table 2.2: Parameters for benchmark functions. $InitialRange$, the variable range in biased initial particles; V_{max}, the max velocity; $Size$, the number of particles; Max_Gen, the max generation; Dim, the number of dimensions. Parameters for functions f_1 to f_5 are set to the same values as in (52) and parameters for functions f_6 to f_{10} are set to the same values as in (30).

- Unified particle swarm optimization (UPSO) (49)
- Fitness-distance-ratio based particle swarm optimization (FDR_PSO) (52)
- Cooperative PSO (CPSO_H)(9)
- Comprehensive learning PSO (CLPSO) (30)
- Our improved PSO (InformPSO)

Graphs (a) to (i) of Figure 2.3 present the results of InformPSO, PSO_w, FDR_PSO and CLPSO on the ten optimization functions introduced in the previous section. From graphs (a) to (e), we note that InformPSO converges to the global optima with a significantly better speed. On functions f_1, f_2 and f_3, PSO_w and CLPSO achieve the best solution within 10^{-20}. They get to 10^{-30} at a cost of convergence speed on functions f_4, f_5. Worse, PSO_w tends to prematurely converge on these two unimodal functions. On functions f_2, f_3 and f_5, the best fitness value of InformPSO keeps changing throughout the search process, reaching 10^{-40} within 1000 iterations. InformPSO steadily converges to global optima (see graphs (b), (c), and (e)). On function f_1, f_4, InformPSO achieves high precision with better convergence speed than other the three PSO variants, especially when it is compared with PSO_w and CLPSO (see graphs (a) and (d)). From graphs (a) to (e), we see that FDR_PSO also performs well on unimodal functions; however, it could not converge as fast as InformPSO.

Function	Best/Worst Fitness Value			
	PSO_w	FDR_PSO	CLPSO	InformPSO
f_1	7.70×10^{-12}	3.72×10^{-33}	6.56×10^{-15}	9.85×10^{-41}
	2.03×10^{-8}	6.72×10^{-32}	1.16×10^{-14}	8.98×10^{-41}
f_2	9.60×10^{-41}	4.72×10^{-41}	4.14×10^{-35}	7.95×10^{-41}
	9.82×10^{-38}	9.30×10^{-41}	1.78×10^{-32}	8.77×10^{-41}
f_3	8.70×10^{-41}	2.90×10^{-41}	1.19×10^{-41}	7.18×10^{-41}
	8.91×10^{-41}	5.30×10^{-41}	3.37×10^{-41}	8.08×10^{-41}
f_4	9.36×10^{-20}	5.72×10^{-33}	8.5×10^{-3}	1.06×10^{-35}
	4.71×10^{-17}	4.13×10^{-30}	0.1092	1.38×10^{-33}
f_5	5.12×10^{-20}	9.20×10^{-41}	7.60×10^{-11}	9.79×10^{-41}
	1.16×10^{-16}	9.71×10^{-41}	3.50×10^{-06}	9.85×10^{-41}

Table 2.3: Results of each algorithm on unimodal functions. Each algorithm was continuously run 30 times on each unimodal function. The fitness value is described by the function value. All results in this table are derived from our experiments.

The best/worst experimental results of each algorithm clearly indicate that InformPSO surpasses the other three variants. It has a large potential search space, and achieves higher function precision in fewer iterations. These good results do not occur in the other PSO variants, because the *gBest* employed in them changes slowly and cannot correctly guide other particles' flying. The reason that InformPSO performs well is because it employs a clonal selection operation for *gBest*, which is deterministic and makes *gBest* implement local evolution so that

InformPSO is able to find the correct direction in a short time and accelerating convergence. CLPSO does not perform well on unimodal function, because a particle in CLPSO learns from other particles' *pBest* instead of *gBest* to maintain population diversity which in turn leads to slow convergence.

From graphs (f) to (j), we observe that InformPSO converges faster and achieves better results, except on function f_9 (Weierstrass), when compared to FDR_PSO. It attains global minima for the three multimodal functions f_6, f_8, and f_9. On multimodal functions, except for function f_7 (Ackley), PSO_w and FDR_PSO perform well in the early iterations. However, they rapidly lose population diversity and easily converge to local optima. And they are unable to improve the best fitness in later iterations, i.e., *gBest* has no ability to get out of local optima when it is trapped in one. From graphs $(f), (h), (i), (j)$, we find that premature convergence does not occur with InformPSO. The better population diversity of InformPSO results from its use of an information function which adjusts particles' velocity variable-wise towards *gBest*, by taking into account their distribution and the number of evolutionary generations, and effectively prevents premature convergence. The clonal selection improves the ability of *gBest* to move out of local optima and makes it implement local evolution. Consequently, the best particle candidate varies so that it correctly guides other particles' movements as well as accelerating convergence.

From graphs $(f), (h), (i), (j)$, we can see that CLPSO achieves a performance relatively close to that of InformPSO. However, from the results of CLPSO and InformPSO run continuously 20 times on the function shown in graph (k), we notice that InformPSO has a better ability to move out of local optima. Although CLPSO achieves a strong population diversity, it is unable to break away from local optima when it is trapped in one. Graph (g) also shows the rapid convergence of InformPSO compared to CLPSO. For functions *Ackley* and $Schwefel$, both algorithms converge to local optima. The reason is that local optima are very far from the global optima in these two functions. The clonal selection of InformPSO did not help it break away from local optima throughout the whole process of iterations and a nonlinear time sequence controls the random drift, so InformPSO is unable to get out of local optima for this kind of function optimization. Also, it can be observed that all PSO variants fail on the functions *Ackley* and *Schwefel*, which are much harder to optimize.

Fun	Best/Worst Fitness Values Achieved					
	PSO_w	PSO_cf_local	FDR_PSO	FIPS	CLPSO	InformPSO
f_6	0.1132	2.80×10^{-2}	0.0295	9.32×10^{-2}	6.71×10^{08}	0
	0.155	6.34×10^{-2}	0.2533	1.31×10^{-1}	7.4×10^{-3}	2.7×10^{-2}
f_7	3.55×10^{-15}	5.78×10^{-2}	7.11×10^{-15}	3.75×10^{-15}	7.11×10^{-15}	8.88×10^{-16}
	3.55×10^{-15}	2.58×10^{-1}	7.11×10^{-15}	2.13×10^{-14}	3.55×10^{-15}	3.55×10^{-15}
f_8	4.9748	3.48	3.9798	1.33	0	0
	10.9445	9.05	8.9546	2.12	0.995	1.20×10^{-8}
f_9	0	5.16×10^{-2}	7.11×10^{-15}	2.02×10^{-3}	0	0
	0.0011	7.85×10^{-2}	2.42×10^{-4}	6.40×10^{-3}	0	0
f_{10}	4.75×10^{2}	2.93×10^{2}	7.10×10^{2}	7.10×10^{1}	1.27×10^{-4}	1.27×10^{-4}
	5.92×10^{2}	8.78×10^{2}	1.11×10^{3}	1.50×10^{2}	1.18×10^{2}	1.18×10^{2}

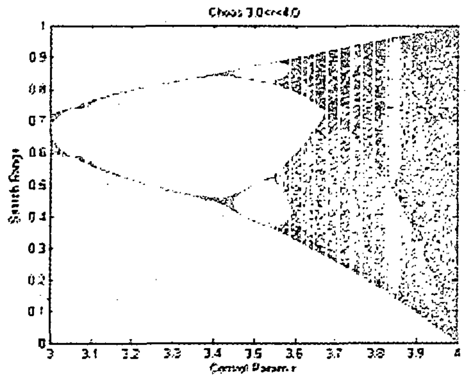
Table 2.4: Results on multimodal functions. Each algorithm was run continuously 20 times on each multimodal function. The best/worst results of PSO_w, PSO_cf_local, FIPS, FDR_PSO, CLPSO and InformPSO are from our experiments, while those of PSO_cf_local and FIPS are from (30).

The results in Table 2.4 further illustrate that InformPSO yields the best precision on the five multimodal functions. It is able to converge to global optima, 0, on functions f_6, f_8 and f_9 in each case except two cases on function f_6. For function *Ackley*, it also reaches a local optimum, 8.8818e-016, better than the other PSO variants. In the experiments on multimodal functions, we can see that PSO_w on function f_9, CLPSO and InformPSO on functions f_6, f_8, f_9, are able to prevent premature convergence and get to global optima. We also notice that InformPSO converges faster, results in better optima, is more robust, and prevents premature convergence more effectively than the other PSO variants.

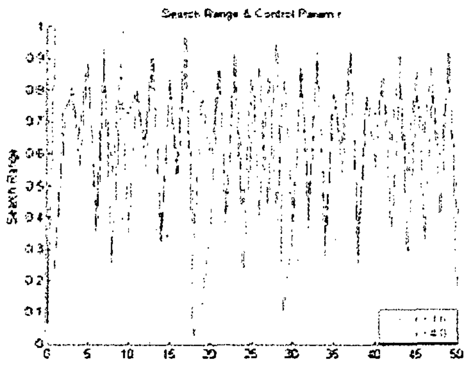
2.5 Conclusions

Following an analysis of the main causes of the premature convergence shown by the original PSO, we propose a novel PSO algorithm, called InformPSO, based on principles of adaptive diffusion and hybrid mutation. On the one hand, inspired by the physics of information diffusion, we design a function to achieve a better particle diversity, by taking into account the distribution of particles and the number of evolutionary generations and by adjusting their "social cognitive" abilities. On the other hand, based on genetic self-organization and chaos evolution, we build clonal selection into InformPSO to implement local evolution of the best particle candidate, $gBest$, and make use of a logistic sequence to control the random drift of $gBest$. These techniques greatly enhance the ability to break away from local optima. The global convergence of the algorithm is proved using the Markov chain theorem. Experiments on optimization of unimodal and multimodal benchmark functions show that, compared with some other PSO variants, InformPSO converges faster, results in better optima, is more robust, and more effectively prevents premature convergence.

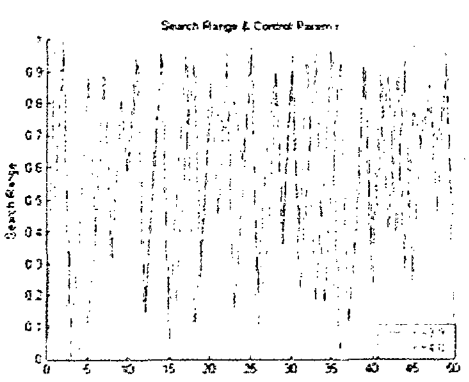
However, although InformPSO achieves higher precision than other PSO variants on the $Schewefel$ and $Ackley$ functions, it also converges to local optima. The reason is that for these two functions where local optima are very far from global optima, the clonal selection operation is a small mutation and unable to work during most of the iterations, and a large mutation guided by a nonlinear time sequence is too random to help $gBest$ get out of local optima. Our work in the future is to broaden the test range and look for new cooperations among particles to improve the algorithm's convergence performance.



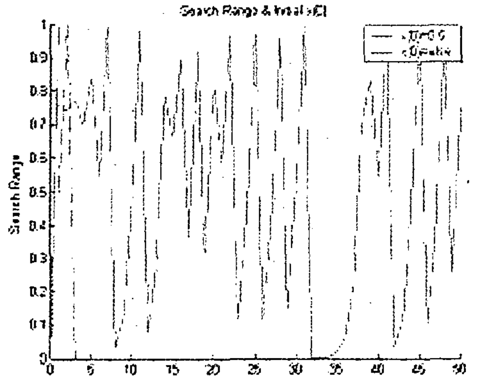
(a) The chaos process of the logistic sequence



(b) Logistic sequence with r=3.6 vs. 4.0

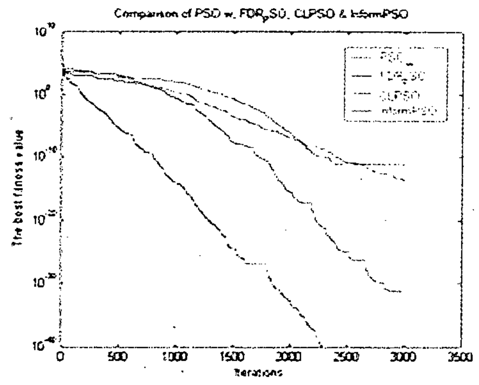


(c) Logistic sequence with r=3.9 vs. 4.0

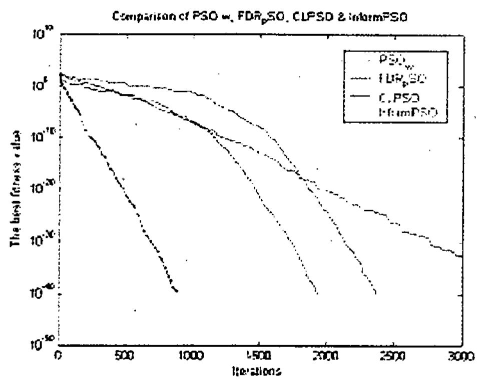


(d) The influence of initial x on the logistic sequence

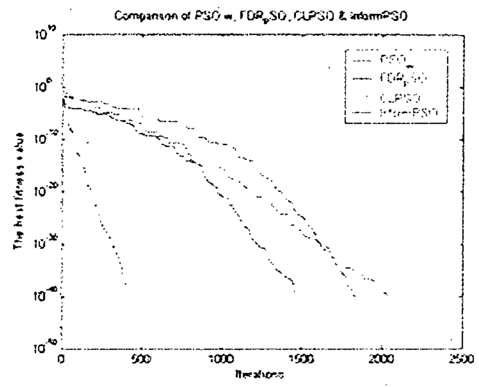
Figure 2.2: Logistic sequence covering range



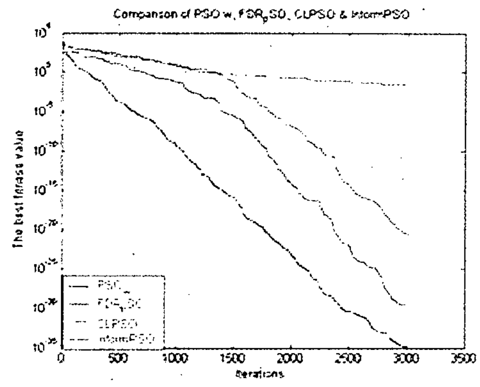
(a) f_1: Sphere



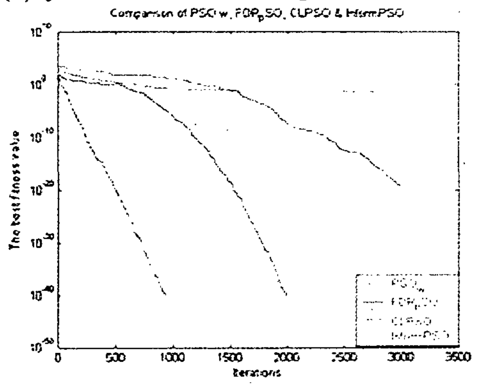
(b) f_2: Hyper-ellipsoid



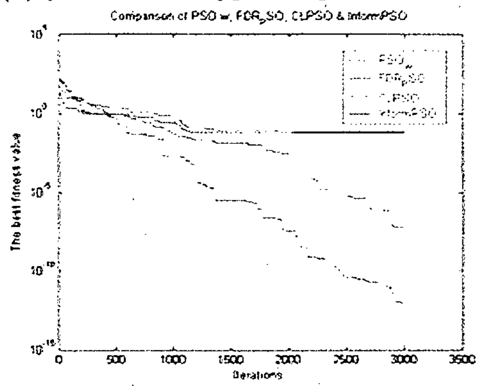
(c) f_3: Sum of different powers



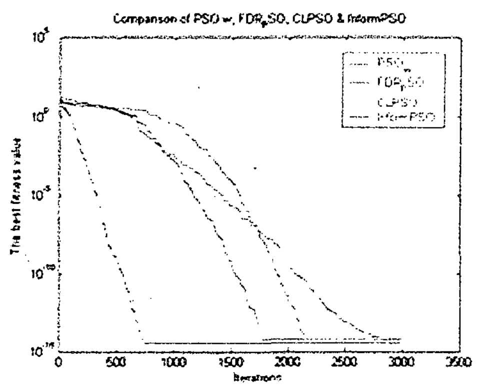
(d) f_4: Rotated hyper-ellipsoid



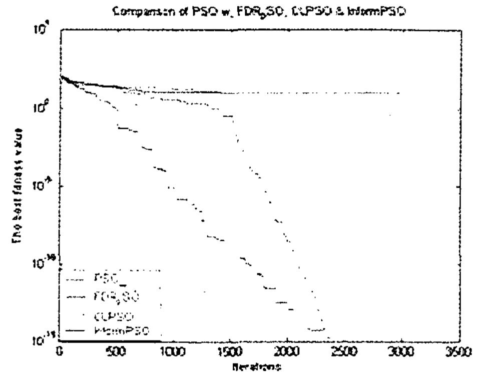
(e) f_5: Rosenbrock



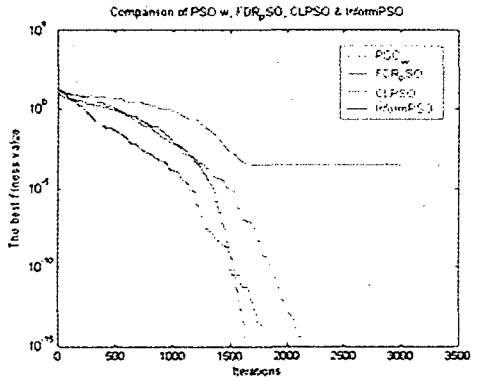
(f) f_6: Griewank



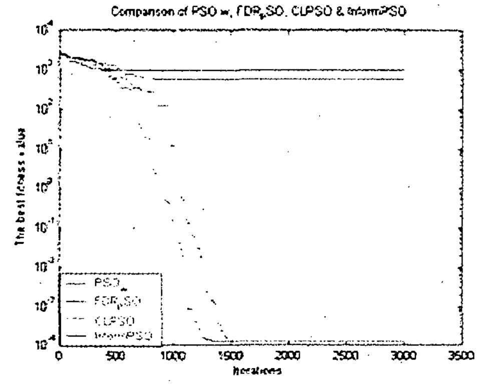
(g) f_7: Ackley



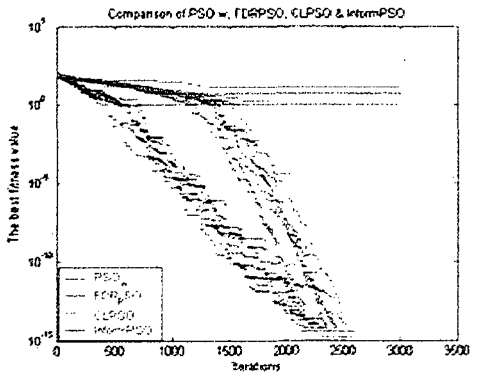
(h) f_8: Rastrigin



(i) f_9: Weierstrass



f_{10}: Schwefel



(k) f_8: Rastrigin

Figure 2.3: Comparison of convergence characteristic. It presents the convergence characteristic of InformPSO vs. PSO_w, FDR_PSO, CLPSO in terms of the best fitness value on benchmark functions. The four algorithms were run continuously for 30 trials on each function. The best fitness values of each generation have been plotted in graphs.

Chapter 3

Particle Swarm Optimizer for the Variable Weighting Problem

In this chapter, we propose a particle swarm optimizer to obtain optimal variable weights in soft projected clustering of high-dimensional data. Good clustering quality is very dependent on a suitable objective function and an efficient search strategy. Our new soft projected clustering algorithm employs a special k-means objective weighting function, which takes the sum of the within-cluster distances of each cluster along relevant dimensions preferentially to irrelevant ones. If a variable is considered insignificant and assigned a small weight, the corresponding distance along this dimension will thus tend to be zero because the weights are handled exponentially. As a result, the function is more sensitive to changes in variable weights. In order to precisely measure the contribution of each dimension to each cluster, we assign a weight to each dimension in each cluster, following (14; 15; 36).

The new algorithm also utilizes a non-normalized representation of variable weights in the objective function and transforms the constrained search space for the variable weighting problem in soft projected clustering into a redundant closed space, which greatly facilitates the search process. Instead of employing local search strategies, the new algorithm makes full use of a particle swarm optimizer to minimize the given objective function. It is simple to implement and the results are much less sensitive to the initial cluster centroids than those of other soft projected clustering algorithms. Experimental results on both synthetic datasets from algorithm generators and real datasets from the UCI database show that it can greatly improve cluster quality.

This chapter is organized as follows. Section 1 reviews some previous related work on the variable weighting problem in soft projected clustering. Section 2 explains the rationale behind using PSO for this problem and utilizes a particle swarm optimizer to assign the optimal variable weight vector for each cluster in soft projected clustering. Simulations on synthetic data are given in Section 3, which contains the description of synthetic datasets with different characteristics, the experimental setting for the PSO, and experimental results. In Section 4, experimental results on three real datasets are presented. An application of our new soft projected clustering algorithm to handle the problem of text clustering is given in Section 5. Section 6 draws conclusions and gives directions for future work.

3.1 Soft Projected Clustering High-dimensional Data

Assuming that the number of clusters k is given and that all of the variables for the datasets are comparable, soft projected clustering algorithms (14; 15; 32; 36) have been designed to discover clusters in the full dimensional space, by assigning a weight to each dimension for each cluster. The variable weighting problem is an important issue in data mining (43). Usually, variables that correlate strongly with a cluster obtain large weights, which means that these variables contribute strongly to the identification of data objects in the cluster. Irrelevant variables in a cluster obtain small weights. Thus, the computation of the membership of a data object in a cluster depends on the variable weights and the cluster centroids. The most relevant variables for each cluster can often be identified by the weights after clustering. The performance of soft projected clustering largely depends on the use of a suitable objective function and an efficient search strategy. The objective function determines the quality of the partitioning, and the search strategy has an impact on whether the optimum of the objective function can be reached. Recently, several soft projected clustering algorithms have been proposed to perform the task of projected clustering and to select relevant variables for clustering. C. Domeniconi et al. proposed the LAC algorithm, which determines the clusters by minimizing the following objective function (14; 15):

$$F_{w,u,z}(w) = \sum_{l=1}^{k}\sum_{j=1}^{m}(w_{l,j} * X_{l,j} + h * w_{l,j} * \log w_{l,j}) \tag{3.1}$$

$$s.t.\{ \begin{array}{ll} \sum_{j=1}^{m} w_{l,j} = 1 & 0 \leq w_{l,j} \leq 1, 1 \leq l \leq k \\ \sum_{l=1}^{k} u_{i,l} = 1 & u_{i,l} \in (0,1), 1 \leq i \leq n \end{array} \tag{3.2}$$

where $X_{l,j} = \frac{\sum_{i=1}^{n} u_{i,l} * (x_{i,j} - z_{l,j})^2}{\sum_{i=1}^{n} u_{i,l}}$, $w \in [0,1]$, $u \in 0,1$, and z represent the variable weight, the cluster membership, and the cluster centroid, respectively. k, n, m are the number of clusters, the number of data objects and the number of dimensions, respectively. Throughout this chapter, we will adopt the same notation. $u_{i,l}$ is the membership of data object i in cluster l. $x_{i,j}$ is the value of data object i on dimension j and $z_{l,j}$ is the centroid of cluster l on dimension j. d is the distance function measuring the similarity between two data objects. $X_{l,j}$ represents the squared variance of cluster l along dimension j. $w_{l,j}$ is a weight assigned to cluster l on dimension j. U, Z, and W represent the cluster membership matrix of data objects, the cluster centroids matrix, and the dimensional weights matrix, respectively.

The LAC algorithm is summarized as follows:

```
Input:   Select $k$ well-scattered data objects as initial centroids;
         Set $w_{l,j} = \frac{1}{m}$ for each dimension in each cluster.

Repeat:  Update the cluster memberships of data objects $U$ by equation 3.3;
         Update the cluster centroids $Z$ by equation 3.4;
         Update the dimension weights $W$ by equation 3.11;

Until:   (the objective function obtains its local minimum value)
```

The LAC algorithm finds a solution that is a local minimum of the above objective function. In the LAC algorithm, U and Z are updated in the same way as in the k-means algorithm. Let $q(t) = \sum_{j=1}^{m} [w_{t,j} * (x_{i,j} - z_{t,j})^2]$, $q^* = \min q(t)$, $1 \leq t \leq k$; then

$$u_{l,i} = \begin{cases} 1 & \text{if } q(t) = q^* \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

$$z_{l,j} = \frac{\sum_{i=1}^{n} u_{l,i} * x_{i,j}}{\sum_{i=1}^{n} u_{l,j}}, for 1 \leq l \leq k and 1 \leq j \leq m \tag{3.4}$$

W is updated according to the following formula:

$$w_{l,j} = \frac{e^{\frac{-X_{l,j}}{h}}}{\sum_{t=1}^{m} e^{\frac{-X_{l,t}}{h}}} \tag{3.5}$$

where h is a parameter that is used to maximize or minimize the influence of X on $w_{l,j}$.

In LAC 2007 (14), the authors employed a new objective function and a weight updating schema. Their work is similar to the work described in EWKM (2007)(36). In LAC 2007(14), the authors also proposed a method based on well-scatter data to initialize the cluster centroids and an ensemble approach to overcome the difficulty of tuning the parameter h. However, their initialization needs to calculate the distance between high-dimensional data in the whole dimension space, whereas the very hypothesis of this work is that this distance is not reliable. It is not clear yet whether LAC is significantly improved by the new initialization schema.

In (32), Joshua Z. Huang et al. proposed another objective function and derived an algorithm called the W-k-means algorithm.

$$F_{w,u,z}(w) = \sum_{l=1}^{k}\sum_{i=1}^{n}\sum_{j=1}^{m}[u_{l,i} * w_j^{\beta} * d(x_{i,j}, z_{l,j})] \tag{3.6}$$

$$s.t.\{ \begin{array}{ll} \sum_{j=1}^{m} w_j = 1 & 0 \le w_j \le 1,\ 1 \le l \le k \\ \sum_{l=1}^{k} u_{i,l} = 1 & u_{i,l} \in (0,1),\ 1 \le i \le n \end{array} \tag{3.7}$$

Here, β is a parameter greater than 1. The objective function is designed to measure the sum of the within-cluster distances along a subset of variables rather than over the entire variable (dimension) space. The variable weights in the W-k-means need to meet the constraints of the objective function in equation 3.7, which is identical to equation 3.2 in LAC. The formula for updating the weights is also different and is written as follows:

$$w_j = \{ \begin{array}{ll} 0 & \text{if } D_j = 0 \\ \frac{1}{\sum_{t=1}^{m}[\frac{D_j}{D_t}]^{\frac{1}{\beta-1}}} & D_j = \sum_{l=1}^{k}\sum_{i=1}^{n}[u_{i,l} * d(x_{i,j}, z_{l,j})] \end{array} \tag{3.8}$$

where the weight w_j for the j^{th} dimension is inversely proportional to the sum of all the within-cluster distances along dimension j. A large weight value corresponds to a small sum of within-cluster distances in a dimension, indicating that the dimension is more important in forming the clusters. The W-k-means algorithm is a direct extension of the k-means algorithm. The W-k-means assigns a weight to each variable and seeks to minimize the sum of all the within-cluster distances in the same subset of variables. Furthermore, updating of variable weights is dependent on the value of the parameter β. In addition, the W-k-means algorithm does not utilize an efficient local search strategy. Consequently, it often has difficulty correctly discovering clusters which are embedded in different subsets of variables. Thus, the algorithm is inappropriate in the case of high-dimensional data where each cluster has its own relevant subset of variables.

Liping Jing et al. in (36) proposed the entropy weighting k-means algorithm (EWKM), which employs a similar iterative process to that used by the W-k-means algorithm. The objective function in the EWKM algorithm takes both the within-cluster dispersion and the weight entropy into consideration. The function is described as follows:

$$F_{w,u,z}(w) = \sum_{l=1}^{k} \sum_{j=1}^{m} (w_{l,j} * X_{l,j} + h * w_{l,j} * \log w_{l,j}) \tag{3.9}$$

$$s.t. \{ \begin{array}{ll} \sum_{j=1}^{m} w_{l,j} = 1 & 0 \leq w_{l,j} \leq 1, 1 \leq l \leq k \\ \sum_{l=1}^{k} u_{i,l} = 1 & u_{i,l} \in (0,1), 1 \leq i \leq n \end{array} \tag{3.10}$$

The constraints in equation 3.10 of the above function are identical to those in the W-k-means algorithm. The formula for updating the weights is written as follows:

$$w_{l,j} = \frac{e^{\frac{-D_{l,j}}{r}}}{\sum_{t=1}^{m} e^{\frac{-D_{l,t}}{r}}}, D_{l,j} = \sum_{i=1}^{n} [u_{i,l} * (x_{i,j} - z_{l,j})^2] \tag{3.11}$$

This objective function depends heavily on the value of parameter r. If r is too small, the entropy section has little effect on the function. On the other hand, if r is too big, the entropy section could have too strong an effect on the function. Therefore, the value of r is empirically determined and application-specific. The computational complexity of LAC, the W-k-means and EWKM is $O(mnkT)$, where T is the total number of iterations and m, n, k are the number of dimensions, the number of data objects and the number of clusters, respectively. Their computational complexity is good and increases linearly as the number of dimensions, the number of data objects or the number of clusters increases. Thus, these three principal algorithms for soft projected clustering can converge to a local minimal solution after a finite number of iterations.

LAC, the W-k-means and EWKM all stem from the k-means algorithm and all three share some common problems. First, the constrained objective functions they provide have their drawbacks, which have been described above. Second, the cluster quality they yield is highly sensitive to the initial cluster centroids (see experiments). These algorithms employ local search strategies to optimize the objective functions with constraints. The local search strategies provide good convergence speed to the iterative process but greatly decrease the clustering quality.

3.2 Particle Swarm Optimizer for Variable Weighting(PSOVW)

In this section, we develop a particle swarm optimizer, called PSOVW, to solve the variable weighting problem in soft subspace clustering for high-dimensional data. We begin by showing how the problem of minimization of a special k-means function with constraints can be transformed into a nonlinear optimization problem with bound constraints, using a non-normalized weight representation technique, and how a particle swarm optimizer can be used to control the search progress. We then give a detailed description of our PSOVW algorithm, which minimizes the transformed problem. Finally, we discuss the computational complexity of the proposed algorithm.

3.2.1 Rationale for using PSO for variable weighting

Basically, the variable weighting problem in soft projected clustering is a continuous nonlinear optimization problem with many equality constraints. It is a difficult problem, and has been researched for decades in the optimization field. Computational intelligence-based techniques, such as the genetic algorithm (GA), particle swarm optimization (PSO) and ant colony optimization (ACO), have been widely used to solve the problem.

GA is a heuristic search technique used in computing to find exact or approximate solutions to optimization problems. Genetic algorithms are a particular class of evolutionary computation that use techniques inspired by evolutionary biology, such as inheritance, mutation, selection, and crossover. In GA, the individuals of the population are usually represented by candidate solutions to an optimization problem. The population starts from randomly generated individuals and evolves towards better solutions over generations. In each generation, the fitness of every individual is evaluated; multiple individuals are selected from the current population based on their fitness, and recombined and mutated to form a new population. The population iterates until the algorithm terminates, by which time a satisfactory solution may be reached.

ACO is a swarm intelligence technique introduced by Marco Dorigo in 1992. It is a probabilistic technique for solving computational problems which can be reduced to finding good paths through graphs. In ACO, ants initially wander randomly and, upon finding food, return to the colony while laying down pheromone trails. Other ants follow the trail, returning and reinforcing it if they find food. However, the pheromone trail starts to evaporate over time, lest the paths found by the first ants should tend to be excessively attractive to the following ones. A short path gets marched over comparatively faster, and thus the pheromone density remains high as it is laid on the path as fast as it can evaporate. Thus, when one ant finds a good (i.e., short) path from the colony to a food source, other ants are more likely to follow that path, and positive feedback eventually results in all the ants following a single path.

PSO is another swarm intelligence technique, first developed by James Kennedy and Russell C. Eberhart in 1995. It was inspired by the social behavior of bird flocking and fish schooling. PSO simulates social behavior. In PSO, each solution to a given problem is regarded as a particle in the search space. Each particle has a position (usually a solution to the problem) and a velocity. The positions are evaluated by a user-defined fitness function. At each iteration, the velocities of the particles are adjusted according to the historical best positions of the population. The particles fly towards better regions through their own effort and in cooperation with other particles. Naturally, the population evolves to an optimal or near-optimal solution.

PSO can generate a high-quality solution within an acceptable calculation time and with a stable convergence characteristic in many problems where most other methods fail to converge. It can thus be effectively applied to various optimization problems. A number of papers have been published in the past few years that focus on the applications of PSO, such as neural network training (61), power and voltage control (27), task assignment (5), single machine total weighted tardiness problems (45) and automated drilling (24).

Moreover, PSO has some advantages over other similar computational intelligence-based techniques such as GA and ACO for solving the variable weighting problem for high-dimensional clustering. For instance:

1. PSO is easier to implement and more computationally efficient than GA and ACO. There are fewer parameters to adjust. PSO only deals with two simple arithmetic operations (addition and multiplication), while GA needs to handle more complex selection and mutation operators.

2. In PSO, every particle remembers its own historic best position, while every ant in ACO needs to track down a series of its own previous positions and individuals in GA have no memory at all. Therefore, a particle in PSO requires less time to calculate its fitness value than an ant in ACO due to the simpler memory mechanism, while sharing the strong memory capability of the ant at the same time. PSO has a more effective memory capability than GA.

3. PSO is more efficient in preserving population diversity to avoid the premature convergence problem. In PSO, all the particles improve themselves using their own previous best positions and are improved by other particles' previous best position by cooperating with each other. In GA, there is "survival of the fittest". The worst individuals are discarded and only the good ones survive. Consequently, the population in GA moves around a subset of the best individuals. ACO easily loses population diversity because ants

are attracted by the largest pheromone trail and there is no direct cooperation among ants. As a result, ACO usually converges to a locally optimal solution.

4. PSO has been found to be robust, especially in solving continuous nonlinear optimization problems, while GA and ACO are good choices for constrained discrete optimization problems. ACO has an advantage over other evolutionary approaches when the graph may change dynamically, and can be run continuously and adapt to changes in real time. Since the PSO method is an excellent optimization methodology for solving complex parameter estimation problems, we have developed a PSO-based algorithm for computing the optimal variable weights in soft projected clustering.

3.2.1.1 PSO for variable weighting (PSOVW)

The objective function

We minimize the following objective function in PSOVW:

$$F_{w,u,z}(w) = \sum_{l=1}^{k}\sum_{i=1}^{n}\sum_{j=1}^{m}[u_{l,i} * (\frac{w_{l,j}}{\sum_{j=1}^{m} w_{l,j}})^{\beta} * d(x_{i,j}, z_{l,j})] \tag{3.12}$$

$$s.t.\{ \begin{array}{l} 0 \leq w_{l,j} \leq 1 \\ \sum_{l=1}^{k} u_{i,l} = 1 \quad u_{i,l} \in \{0,1\},\ 1 \leq i \leq n \end{array} \tag{3.13}$$

Actually, the objective function in equation 3.12 is a generalization of some existing objective functions.

1. If β=0, function 3.12 is similar to the objective function used in the k-means. They differ solely in the representation of variable weights and the constraints.

2. If β=1, function 3.12 is also similar to the first section of the objective function in EWKM, differing only in the representation of variable weights and the constraints.

3. If $w_{l,j} = w_j, \forall l$, function 3.12 is similar to the objective function in the W-k-means, which assigns a single variable weight vector, substituting different vectors for clusters. Again, the only difference is in the representation of variable weights and the constraints.

In PSOVW, β is a user-defined parameter. PSOVW works with all non-negative values of β. In practice, we suggest setting β to a large value (empirically around 10.0). A large value of β makes the objective function more sensitive to changes in weight values. It tends to magnify the influence of those variables with large weights on the within-cluster variance, allowing them to play a strong role in discriminating between relevant and irrelevant variables (dimensions).

In the existing algorithms, a $k * m$ variable weight matrix W, where a weight is a real number assigned to each dimension for each cluster, is usually a direct solution to the objective function and should simultaneously meet the equality constraints that the weights on each row should be normalized to 1. However, the greater the number of clusters, the greater the number of constraints and the harder it is to optimize the corresponding function. In PSOVW, a non-normalized matrix W is employed in the objective function, replacing the normalized matrix. Although the non-normalized representation in the constrained objective function 3.12 is redundant, the constraints can be loosened without affecting the final solution. As a result, the optimization procedure only needs to deal with bound constraints instead of many more complicated equality constraints and it is greatly simplified. To sum up, the weighting schema is developed in a suitable way for the application of PSO.

Initialization

In the PSOVW algorithm, we initialize three swarms.

- The position swarm W of variable weights, which are set to random numbers uniformly distributed in a certain range R.

- The velocity swarm V of variable weights, which are set to random numbers uniformly distributed in the range $[-maxv, maxv]$. Here, $maxv$ means the maximum flying velocity of particles and $maxv = 0.25 * R$.

- The swarm of cluster centroids Z, which are k different data objects randomly chosen out of all the data objects. In all three swarms, an individual is a $k * m$ matrix. Actually, only the velocity swarm V is an extra swarm not included in k-means-type soft projected clustering algorithms.

Update of cluster centroids and partitioning of data objects

Given the variable weight matrix and the cluster centroids, the cluster membership of each data object is calculated as follows.

Let

$q(t) = \sum_{j=1}^{m} [(\frac{w_{i,j}}{\sum_{j=1}^{m} w_{l,j}})^{\beta} * d(x_{i,j}, z_{l,j})]$, $q^* = \min q(t)$, $1 \leq t \leq k$,

Then

$$u_i = \left\{ \begin{array}{ll} 1 & \text{if } q(t) = q^* \\ 0 & \text{otherwise} \end{array} \right. \quad (3.14)$$

Once the cluster membership is obtained, the cluster centroids are updated by

$$z_{l,j} = \frac{\sum_{i=1}^{n} u_{l,i} * x_{i,j}}{\sum_{i=1}^{n} u_{l,j}}, for 1 \leq l \leq k and 1 \leq j \leq m \quad (3.15)$$

In our implementation, if an empty cluster results from the membership update by formula 3.14, we randomly select a data object out of the data set to reinitialize the centroid of the cluster.

Crossover learning

Given a value for β, two extra swarms are kept to guide all the particles' movement in the search space. One is the personal best position swarm of variable weights $pBest$, which keeps the best position of the weight matrix W; i.e., $pBest$ will be replaced by W if $F(W) < F(pBest)$. The other is the crossover best position swarm of variable weights $CpBest$, which guides particles to move towards better regions (30). *CpBest* is obtained by a crossover operation between its own *pBest* and one of the other best personal positions in the swarm, represented here by s_pBest.

The tournament mechanism is employed to select *s_pBest*, with the consideration that a particle learns from a good exemplar. First, two individuals, $pBest_1$ and $pBest_2$, are randomly selected. Then their objective function values are compared: $s\_pBest = pBest_1$ if $F(pBest_1) < F(pBest_2)$, $s\_pBest = pBest_2$ otherwise. The crossover is done as follows. Each element at a position of *CpBest* is assigned the value either from *s_pBest* or from *pBest* at the corresponding position. This assignment is made randomly according to a user-defined probability value P_c (see the section Synthetic Data Simulations for further discussion). If, despite the random assignment process, all the elements of *CpBest* take values from its own *pBest*, one element of *CpBest* will be randomly selected and its value will be replaced by the value of the corresponding position from *s_pBest*.

The PSOVW algorithm can be summarized as follows:

```
Initialization:  Randomly initialize the position and velocity swarms, $W$ and $V$.
                 Partition the data objects.
                 Evaluate the fitness of $W$ by the function 3.12.
                 Record the swarm pBest and the best position gBest.

Repeat:          Produce CpBest from pBest for each particle.
                 Update $V$ and $W$ by the equations (2.2) and (4.6) , respectively.
                 If $W$ is in the range of [0, 1], evaluate its fitness by 3.12
                 and update pBest and gBest.

Until:           (the objective function reaches a global minimum value,
                 or the number of function evaluations reaches the maximum threshold.)
```

3.2.2 Computational complexity

The computational complexity can be analyzed as follows. In the PSOVW algorithm, the *CpBest*, position and velocity of a particle are $k * m$ matrices.

In the main loop procedure, after initialization of the position and velocity, generating *CpBest* and updating position and velocity need $O(mk)$ operations for each particle. Given the weight matrix W and the cluster centroids matrix Z, the complexity for partitioning the data objects is $O(mnk)$ for each particle. The complexity for updating cluster centers is $O(mk)$ for each particle. Assuming that the PSOVW algorithm needs T iterations to converge, the total computational complexity of the PSOVW algorithm is $O(smnkT)$. The swarm size s is usually a constant set by the user. So, the computational complexity still increases linearly as the number of dimensions, the number of objects or the number of clusters increases.

3.3 Synthetic Data Simulations

A comprehensive performance study has been conducted to evaluate the performance of PSOVW on high-dimensional synthetic datasets as well as on real datasets. Three of the most widely used soft projected clustering algorithms, LAC (14), the W-k-means (32) and EWKM (36), were chosen for comparison, as well as the basic k-means [1]. The goal of the experiments is to assess the accuracy and the efficiency of PSOVW. We compared the clustering accuracy of the five algorithms, examined the variance of the cluster results and measured the running time spent by these five algorithms. These experiments provide information on how well and how fast each algorithm is able to retrieve known clusters in some subspaces of a high-dimensional data space and how sensitive each algorithm is to the initial centroids. In this section, we describe the synthetic datasets and the results.

[1]The k-means is included in order to show that projected clustering algorithms are necessary on high-dimensional data sets.

3.3.1 Synthetic datasets

We generate high-dimensional datasets with clusters embedded in different subspaces using the synthetic data generator from Zait and Messatfa (67). The synthetic data generator has often been used to investigate the performance of subspace clustering algorithms. The data generator allows us to take control of characteristics of the data set such as cluster structures and the percentage of overlapped relevant dimensions among clusters. It also facilitates measuring the cluster accuracy of each algorithm by comparing the result of the algorithms to the known clusters.

We conducted experiments using the above four algorithms together with the PSOVW algorithm on synthetic datasets. In these synthetic datasets, different clusters have their own relevant dimensions, which can overlap (36). The data values are normally distributed on the relevant dimensions of a cluster and the range of mean values is specified by the range [0, 100]. Random positive numbers are on the irrelevant dimensions and random values are uniformly distributed in the range [0, 10].

Two types of synthetic datasets are generated. They differ in the variance of clusters on their relevant dimensions. In type 1, we assign the same variance 0.9 to each relevant dimension in a cluster. In type 2, each relevant dimension is randomly assigned a variance and the cluster variances are randomly selected from the range [1, 10]. We generated a total of 16 synthetic datasets with varying numbers of dimensions and varying numbers of clusters. For each type of data, there are eight datasets, each of which has 4 or 10 clusters, combined with 20, 100, 1000 or 2000 dimensions. The datasets with 4 clusters have 50 data objects in a cluster, the clusters are well-separated and the percentage of overlapped relevant dimensions of clusters is low (near 0). The datasets with 10 clusters are more complicated: each cluster has 50 data objects and relevant dimensions of clusters are heavily overlapped (around 0.8).

3.3.2 Parameter settings for algorithms

The Euclidean distance is used to measure the dissimilarity between two data objects for synthetic and real datasets. Here, we set $\beta = 10$. The maximum number of function evaluations is set to 2,000 over all synthetic datasets in PSOVW and 500 in the other four algorithms. Given an objective function, LAC, the W-k-means and EWKM quickly converge to local optima after a few iterations, while PSOVW needs more iterations to get to its best optima. There are four additional parameters in PSOVW that need to be specified. They are the swarm size s, the inertia weight w, the acceleration constant $c1$ and the learning probability P_c. Actually, the four extra parameters have been fully studied in the particle swarm optimization field and we set the parameters in PSOVW to the values used in (30). s is set to 10. w is linearly decreased from 0.9 to 0.7 during the iterative procedure, and $c1$ is set to 1.49445. In order to obtain a better population diversity, particles in PSOVW are required to have different exploration and exploitation ability. So, P_c is empirically set to $P_c = 0.5 * \frac{e^{f(i)} - e^{f(1)}}{e^{f(s)} - e^{f(1)}}$, where $f(i) = 5 * \frac{i}{s-i}$for each particle. The values of learning probability P_c in PSOVW range from 0.05 to 0.5 as for CLPSO (30). The parameter r in EWKM is set to 1.

3.3.3 Results for synthetic data

To test the clustering performance of PSOVW, we first present the clustering results of the five algorithms (k-means, LAC[1], W-k-means, EWKM and PSOVW). Then we employ the PSO method with different functions and report the clustering results yielded by them. This provides further evidence for the superior effectiveness of PSOVW in clustering high-dimensional data. Finally, we examine

[1]Since it is difficult to simulate the function combining different weighting vectors in the latest LAC algorithm (14), we tested its performance with the values from 1 to 11 for the parameter $1/h$, as the authors did in (14), and report the best clustering accuracy on each trial.

the average clustering accuracy, the variance of the clustering results and the running time of the five algorithms using synthetic data sets, and make an extensive comparison between them. The clustering accuracy is calculated as follows:

$$ClusteringAccuracy = \sum_{l=1}^{k} \frac{N_l}{n}$$

where N_l is the number of data objects that are correctly identified in the genuine cluster l and n is the number of data objects in the data set. The clustering accuracy is the percentage of the data objects that are correctly recovered by an algorithm.

In order to compare the clustering performance of PSOVW with those of the k-means, LAC, the W-k-means and EWKM, we run these five algorithms on datasets of each type. Since PSOVW was randomly initialized and the other algorithms are sensitive to the initial cluster centroids to a certain degree, we run each algorithm for ten trials on each dataset and record the average clustering accuracy that each algorithm achieved in Table 3.1.

A number of observations can be made by analyzing the results in Table 3.1. First, PSOVW seems to perform much better than the other algorithms tested on both types of generated data sets. On average, it surpasses the W-k-means, EWKM and LAC as well as the k-means on the 16 synthetic subspace datasets. It correctly recovers clusters embedded in different variable subspaces of high-dimensional data and achieves 100 percent clustering accuracy on most of trials. The reason is that the synthetic datasets are not noisy. Noise-free datasets are appropriate for the purpose of this experiment. In particular, due to the PSO search strategy, PSOVW totally and correctly recovers clusters over the datasets with 4 well-separated clusters on each trial regardless of the shapes of these clusters and the variances of clusters along relevant dimensions. However, PSOVW occasionally misses an entire cluster on the more complicated, 10-cluster datasets. It sometimes fails to differentiate between two very close groups, whose relevant dimensions overlap considerably, and merges them into one cluster. For

Datasets	k	m	k-means	LAC	W-k-means	EWKM	PSOVW
Type 1	4	20	68	70.3	86	77.4	**100**
		100	79.4	73	85.8	64.4	**100**
		1000	70.5	71.1	89.2	78.8	**100**
		2000	73.6	85.9	79.7	84.6	**100**
	10	20	61.3	74.5	75.1	74.5	**90.9**
		100	34.4	68.4	72.2	69.4	**94**
		1000	33.4	70.4	76.8	67	**91.2**
		2000	39.7	68.4	75.8	69.8	**86.8**
Type 2	4	20	63.1	83.4	78.2	89.3	**100**
		100	59.9	78.7	79	71.9	**100**
		1000	73.9	78.3	81.5	76.1	**100**
		2000	78.2	71.9	94.2	83.9	**100**
	10	20	34.4	66.1	77.7	63.9	**92.5**
		100	27.3	63	73.7	67	**94**
		1000	32.4	69.9	76.9	72.7	**88.2**
		2000	24	71.1	63.6	71	**90.3**

Table 3.1: Average clustering accuracy of PSOVW, k-means, W-k-means, EWKM and LAC over ten trials on each dataset.

example, the relevant dimensions of one cluster in the Type 1 dataset with 10 clusters and 20 dimensions are $\{1\ 3\ 5\ 8\ 11\}$ and those of another cluster in the same dataset are $\{1\ 3\ 5\ 11\ 17\}$.

The W-k-means performs the next best on 12 out of 16 datasets. However, from the experimental results in Table 3.1, we cannot see much difference between the performance of the W-k-means and those of LAC and EWKM on Type 2 datasets.

The k-means performs the least well on complicated datasets, such as the datasets with 10 complicated clusters. We cannot see much difference between the results yielded by the k-means and those of LAC and EWKM on the Type 1 datasets with 4 clusters. However, the clustering accuracy of the k-means drops quickly as the variances of clusters along relevant dimensions increase. What is more, the k-means achieved very low clustering accuracy on complicated datasets, such as datasets with 10 clusters. The reason is that the k-means is not a subspace clustering algorithm, so that it fails to handle high-dimensional data due to the equidistance of such data in the whole dimensional space. The results of the k-means in Table 3.1 confirm that it was inferior to soft projected clustering algorithms in complex high-dimensional data clustering.

In order to further investigate the reasons why PSOVW achieves better performance than other algorithms on generated high-dimensional datasets, we also implemented the PSO method with different objective functions presented in soft projected clustering algorithms, namely formula 3.1 in LAC, formula 3.6 in W-k-means, formula 3.9 in EWKM and formula 3.12 in PSOVW. PSO is a heuristic global search strategy, so it can be also employed to optimize other objective functions. Since PSO was randomly initialized, the PSO method with each function was run for 20 trials on each dataset. k and m represent the number of clusters and the number of dimensions, respectively. The average clustering accuracy yielded by the PSO method with each function is recorded in Table 3.2.

Datasets	k	m	Function 3.12 in PSOVW	Function 3.1 in LAC	Function 3.6 in W-k-means	Function 3.9 in EWKM
Type 1	10	20	**90.3**	87.2	89.4	86.5
		100	**93.6**	89.5	85.7	76.7
		1000	**91.9**	90.8	84.4	70.6
		2000	**87**	85.9	83.6	80.1
Type 2	10	20	93.2	**97.4**	92.4	85.9
		100	**92.3**	84.8	85.6	82.8
		1000	**87.6**	82.7	82.9	77
		2000	89.4	**90.8**	84.5	80.5

Table 3.2: Average clustering accuracy of the PSO method with different functions over 20 trials on datasets with 10 clusters.

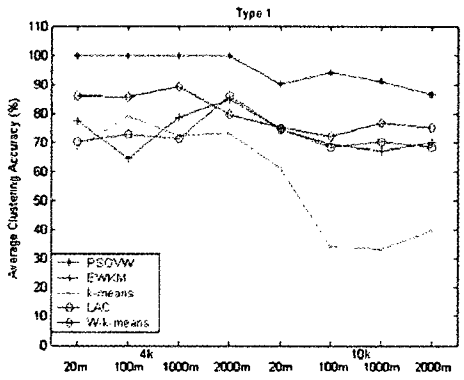
Table 3.2 gives the clustering results of the PSO method with different functions on the datasets with 10 clusters. In our experiments, we discovered that the

PSO method with each of the four different objective functions performs pretty well on the datasets with 4 clusters and achieves 100 percent clustering accuracy on each trial. The reason is that clusters in these datasets are well-separated and the percentage of overlapped relevant dimensions is relatively low, although the clusters are different shapes. Furthermore, we found out that the high clustering accuracy yielded by the PSO method with different functions benefits greatly from the non-normalized representation of variable weights presented in our objective functions, although we also tried several existing techniques to initialize feasible particles. For a fair comparison, we implemented the same representation as in Function 3.12 of PSOVW, where variable weights are only required to meet bound constraints, in the LAC, W-k-means and EWKM functions.

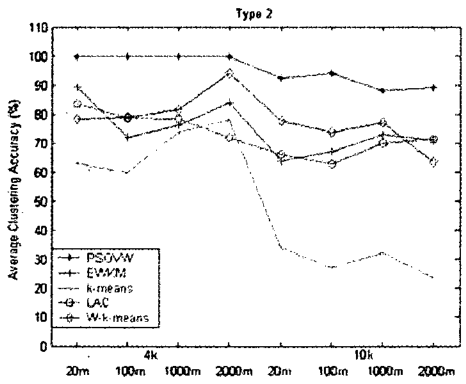
From Table 3.2, we can see that on average, except for the Type 2 datasets with 20 dimensions and 2000 dimensions, PSO with Function 3.12 always performs best on the synthetic datasets with 10 clusters, although PSO with LAC Function 3.1 is very close. The fact that Function 3.6 is less efficient than Functions 3.1 and 3.12 is understandable because it uses one weight per dimension; nevertheless, it performs better than Function 3.9. Obviously, the PSO method greatly improves the performances of LAC, the W-k-means and EWKM. The results reported in both Table 3.1 and Table 3.2 suggest that the good performance of PSOVW is due partly to the PSO method, and partly to the representation of our improved objective function.

Figure 3.1 illustrates the clustering results achieved by each algorithm on each type of dataset, for different numbers of clusters and different numbers of dimensions. The horizontal axis represents the datasets. The vertical axis in (a) and (b) represents the average clustering accuracy value over ten runs on each dataset, while that in (c) and (d) represents the variance of the clustering accuracies yielded by each algorithm in ten runs on each dataset. Graphs (a) and (b) in Figure 3.1 are a graphic depiction of Table 3.1.

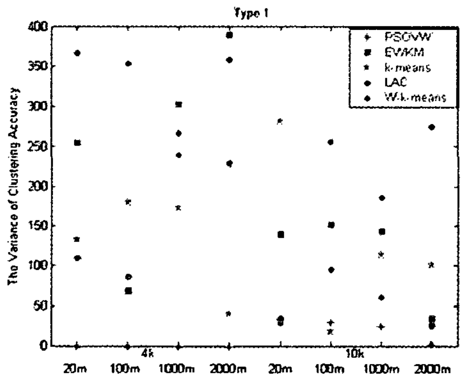
From Figure 3.1 (a) and (b), we notice that the performance of each algorithm drops on the datasets with 10 clusters. However, PSOVW still yields the best



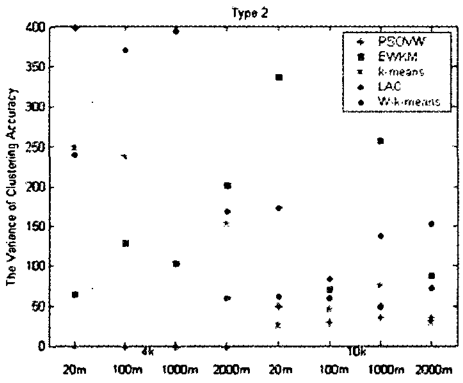
(a) Average cluster accuracy on Type 1



(b) Average cluster accuracy on Type 2



(c) Variance of cluster accuracy on Type 1



(d) Variance of cluster accuracy on Type 2

Figure 3.1: Average clustering accuracy and corresponding variance of PSOVW, *k*-means, W-*k*-means, EWKM and LAC on each dataset.

results of the five algorithms. The performances of the W-*k*-means, EWKM and LAC also seem to remain quite high in clustering accuracy as the datasets become more complicated. However, we find that their overall clustering abilities are very close, and inferior to that of PSOVW. The performance of the *k*-means deteriorated considerably on datasets with 10 clusters. The reason is that PSOVW, the W-*k*-means, EWKM and LAC are soft projected clustering algorithms, which calculate the distance between two data objects in the weighted whole dimensional space so that the clusters of high-dimensional data can be separated from each other. As a result, they were superior to the k-means in handling complex high-dimensional data clustering.

In order to see how sensitive the clustering results of these five algorithms are to the initial cluster centroids, we also examined the variance of ten clustering results of each algorithm on each dataset. From graphs (c) and (d), it can be readily seen that PSOVW yields the least variance in clustering accuracy on most of the datasets. The k-means achieved lower variance in clustering accuracy than PSOVW on some datasets with 10 clusters, because the clustering accuracy yielded by the k-means on these dataset is always very low. The other three soft projected clustering algorithms occasionally work well, achieving low variance in clustering accuracy on some datasets, but their variance is often high. What's more, it is difficult to tell for which type of datasets they are less sensitive to the initial centroids. From graphs (c) and (d) in Figure 3.1, we conclude that PSOVW was far less sensitive to the initial centroids than the other four algorithms.

Table 3.3 presents the average running time of each algorithm on the datasets, with varying k and m. All times are in seconds.

Datasets		Running Time (s)				
k	m	k-means	LAC	W-k-means	EWKM	PSOVW
4	20	**0.094**	1.531	2.031	1.234	7.828
	100	**0.453**	7.500	10.125	6.641	39.156
	1000	**4.422**	74.437	100.343	60.718	206.140
	2000	**8.843**	131.327	200.858	122.686	370.529
10	20	**0.172**	3.000	4.890	2.641	49.172
	100	**0.766**	14.875	24.125	13.203	92.358
	1000	**7.500**	149.249	257.467	132.749	408.451
	2000	**14.984**	298.826	514.684	264.186	700.902

Table 3.3: Average running time of PSOVW vs. k-means, LAC, W-k-means, and EWKM.

From Table 3.3, it is clear that the k-means runs faster than the other four soft projected clustering algorithms. Although LAC, EWKM and the W-k-means all extend the k-means clustering process by adding an additional step in each iteration to compute variable weights for each cluster, LAC and EWKM are much faster algorithms than the W-k-means on datasets. The W-k-means required more time to identify clusters, because its objective function involves a much more complicated computation due to the high exponent. Based on the computational complexities of the various algorithms and the maximum number of function evaluations set in each algorithm, PSOVW was supposed to need 10 40 times the running time the W-k-means requires. In fact, it required only slightly more than ten times the running time that the W-k-means spent on the datasets with 20 and 100 dimensions, and around two times what the W-k-means spent on those with 1000 and 2000 dimensions. This is because the update of weights is simpler in PSOVW than in the other three soft projected clustering algorithms and particles are evaluated only if they are in the feasible closed space. We thus conclude that the much more complicated search strategy of PSOVW allows it to achieve the best clustering accuracy, at an acceptable cost in terms of running time.

Table 3.4 presents the comparison of PSO with the best result of LAC, EWKM and the W-k-means on multiple independent trials.

Since PSOVW uses more resources than LAC, EWKM and the W-k-means, to ensure a very fair comparison of PSOVW with these three soft projected clustering algorithms, we independently ran LAC, EWKM and the W-k-means multiple times on each dataset and compared the best result out of the iterative trials with that of PSOVW. According to the average running time of each algorithm reported in Table 3.3, we ran PSOVW once on each dataset. On the datasets with 20 and 100 dimensions, LAC and EWKM were run 10 times and the W-k-means was run 4 times. On the datasets with 1000 and 2000 dimensions, LAC and EWKM were run 4 times and the W-k-means was run 2 times. The experimental results in Figure 4 still show that PSOVW surpasses the other three algorithms on each data set except on the Type 2 datasets with 10 clusters and 20 and 2000 dimensions. This is because the particles in PSOVW do not work independently

Datasets	Type 1							
	4 k				10 k			
Dimensions	20 m	100 m	1000 m	2000 m	20 m	100 m	1000 m	2000 m
LAC	89	93.5	89.5	**100**	83	92.5	81	76
W-k-means	**100**	**100**	**100**	**100**	85	85	86.5	**85.5**
EWKM	**100**	71.5	95.5	**100**	81	86	76	77
PSOVW	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**85.5**
Datasets	Type 2							
	4 k				10 k			
	20 m	100 m	1000 m	2000 m	20 m	100 m	1000 m	2000 m
LAC	**100**	**100**	**100**	83.5	**88.5**	79	71.5	88.5
W-k-means	**100**	**100**	**100**	**100**	86.5	85	89.5	72.5
EWKM	**100**	**100**	91	96	76	80	**100**	83
PSOVW	**100**	**100**	**100**	**100**	87	**100**	86.5	**100**

Table 3.4: Comparison of PSOVW vs. LAC, W-k-means and EWKM.

but cooperate with each other in order to move to better regions. The quality of the clustering is of prime importance and therefore the time taken to obtain it is at most secondary in many clustering applications. To sum up, the much more complicated search strategy allows PSOVW to achieve the best clustering accuracy, at a cost of longer running time, but the running time is acceptable.

3.4 Test on Real Data

3.4.1 Real-life datasets

A comprehensive performance study was conducted to evaluate our method on real datasets. For the experiments described in this section, we chose three real datasets: the *glass identification* dataset (19), the *Wisconsin breast cancer* dataset (44) and the *insurance company* dataset (62), which were obtained from the UCI database (2). The *glass identification* dataset consists of 214 instances. Each record has nine numerical variables. The records are classified into two classes: 163 window glass instances and 51 non-window glass instances. *Wisconsin breast cancer* data has 569 instances with 30 continuous variables. Each record is labeled as benign or malignant. The dataset contains 357 benign records and 212 malignant records. The *insurance company* dataset has 4000 instances. Each record has 85 numerical variables containing information on customers of an insurance company. A classification label indicating whether the customer is or is not an insurance policy purchaser is provided with each record. The numbers of purchasers and non-purchasers are 3762 and 238, respectively.

3.4.2 Results for real data

To study the effect of the initial cluster centroids, we ran each algorithm ten times on each dataset. Figure 3.2 presents the average clustering accuracy and the corresponding variance of each algorithm on each real dataset. The horizontal axis represents real datasets. In the left graph, the vertical axis represents the average value of the clustering accuracy for each dataset, and in the right graph it represents the variance of the clustering quality.

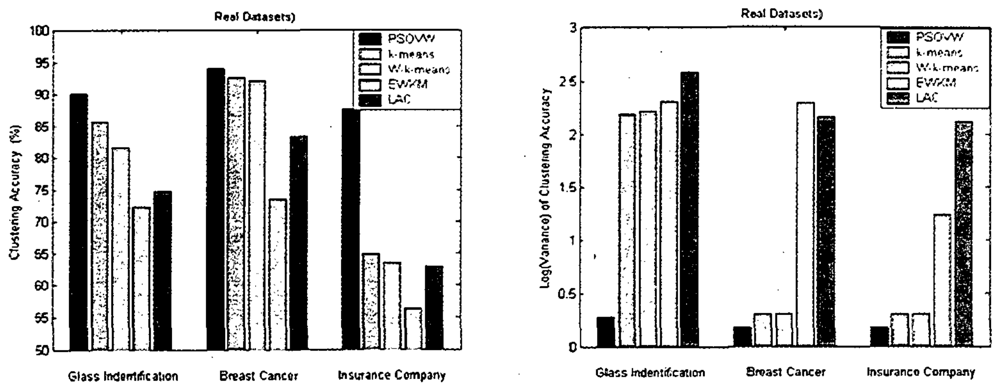


Figure 3.2: Average clustering accuracy and the log variance of the clustering accuracy for each algorithm on each real-life dataset.

From Figure 3.2, we can see that PSOVW outperforms the other algorithms. The average clustering accuracy achieved by PSOVW is high on each dataset, while the corresponding variance is very low. PSOVW is a random swarm intelligent algorithm; however, its outputs are very stable. In our experiments, we also observed that other algorithms can achieve good clustering accuracy on the first two datasets on some runs. However, the performances of LAC and EWKM are very unstable on the glass and breast cancer datasets and the performances of the k-means and W-k-means are unstable on the glass dataset. Their average clustering accuracy is not good, because their iterative procedures are derived from the k-means and are very sensitive to the initial cluster centroids. To sum up, PSOVW shows more capability to cluster these datasets and is more stable than other algorithms.

3.5 PSOVW Application in Text Clustering

3.5.1 Modifications of PSOVW

To further demonstrate the efficiency of PSOVW on real data, we apply it to the problem of text clustering, called Text Clustering via Particle Swarm Optimization (TCPSO). To extend PSOVW to text clustering, a few modifications to the algorithm are necessary. In particular, we have adopted the conventional $tf - idf$ to represent a text document, and make use of the extended Jaccard coefficient, rather than the Euclidean distance, to measure the similarity between two text documents.

Throughout this section, we will use the symbols n, m and k to denote the number of text documents, the number of terms and the number of clusters, respectively. We will use the symbol L to denote the text set, $L_1, L_2, ., L_k$ to denote each one of the k categories, and $n_1, n_2, , n_k$ to denote the sizes of the corresponding clusters. In TCPSO, each document is considered as a vector in the term-space. We used the $tf - idf$ term weighting model (55), in which each document can be represented as:

$$[tf_1 * \log(n/df_1), tf_2 * \log(n/df_2), \cdots, tf_m * \log(n/df_m)] \qquad (3.16)$$

where tf_i is the frequency of the i^{th} term in the document and df_i is the number of documents that contain the i^{th} term. The extended Jaccard coefficient (57) used to measure the similarity between two documents is defined by the following equation:

$$similarity(x,z) = \frac{x \cdot z}{\|x\|^2 + \|z\|^2 - x \cdot z}$$

where x represents one document vector, z represents its cluster centroid vector and the symbol $'\cdot'$ represents the inner product of two vectors. This measure becomes one if two documents are identical, and zero if there is nothing in common between them (i.e., the vectors are orthogonal to each other).

Based on the extended Jaccard coefficient, the weighted similarity between two text documents can be represented as follows:

$$\frac{(w^\beta \cdot x) \cdot (w^\beta \cdot z)}{\|w^\beta \cdot x\|^2 + \|w^\beta \cdot z\|^2 - (w^\beta \cdot x) \cdot (w^\beta \cdot z)}$$

where w is the weight vector for the corresponding cluster whose centroid is z. Therefore, the objective function in TCPSO serves to maximize the overall similarity of documents in a cluster and can be modified as follows:

$$\max F(W) = \sum_{l=1}^{k} \sum_{i=1}^{n} [u_{l,i} \cdot \frac{x \cdot z}{\|x\|^2 + \|z\|^2 - x \cdot z}]$$

where u is the cluster membership of the text document x.

3.5.2 Text datasets

To demonstrate the effectiveness of TCPSO in text clustering, we first extract four different structured datasets from 20 *newsgroups* (2). The vocabulary, the documents and their corresponding key terms as well as the dataset are available in (3). We used the processed version of the 20 news-by-date data set, which is easy to read into Matlab, as a matrix. We also selected four large text datasets from the CLUTO clustering toolkit (1). For all datasets, the common words were eliminated using stop-list and all the words were stemmed using Porter's suffix-stripping algorithm (50). Moreover, any terms that occur in fewer than two documents were removed.

We chose three famous clustering algorithms (Bisection k-means, Agglomeration and Graph) and one soft projected clustering algorithm (EWKM) for comparison with TCPSO. For soft projected clustering algorithms, the similarity between two documents is measured by the extended Jaccard coefficient, while the cosine function is used for the other algorithms, namely Bisection k-means, Agglomeration, Graph-based and k-means. The second group of text datasets as well as the three clustering algorithms, Bisection k-means, Agglomeration and Graph, can be downloaded from the website of CLUTO.

Table 3.5 presents four different structured text datasets extracted from 20 *newsgroups*. Each dataset consists of 2 or 4 categories. n_i is the number of documents randomly selected from each category. m and k represent the number of terms and the number of categories, respectively. The categories in datasets 2 and 4 are very closely related to each other, i.e., the relevant dimensions of different categories overlap considerably, while the categories in datasets 1 and 3 are highly unrelated, i.e., different categories do not share many relevant dimensions. Dataset 2 contains different numbers of documents in each category.

Table 3.6 summarizes the characteristics of four large text datasets from CLUTO. To ensure diversity in the datasets, we selected them from different sources. n, m and k are the number of documents, the number of terms and the number of categories, respectively.

Text datasets $tr41$ and $tr45$ are derived from TREC collections (4). The categories correspond to the documents relevant to particular queries. Text datasets wap and $k1b$ are from the webACE project (11; 17), where each document corresponds to a web page listed in the subject hierarchy of Yahoo.

Dataset	Source Category	n_i	m	k
1	comp.graphics	100	341	2
	alt.atheism	100		
2	talk.politics.mideast	100	690	2
	talk.politics.misc	80		
3	comp.graphics	100	434	4
	rec.sport.hockey	100		
	sci.crypt	100		
	talk.religion.misc	100		
4	alt.atheism	100	387	4
	rec.sport.baseball	100		
	talk.politics.guns	100		
	talk.politics.misc	100		

Table 3.5: The four text datasets extracted from 20 *newsgroups*.

Datasets	Source	n	m	k
5: tr41	TREC	878	7454	10
6: tr45	TREC	927	10128	7
7: wap	webACE	1560	8460	20
8: klb	webACE	2340	13879	6

Table 3.6: The four large text datasets selected from CLUTO.

3.5.3 Experimental metrics

The quality of a clustering solution is determined by two different metrics that examine the class labels of the documents assigned to each cluster. In the following definitions, we assume that a data set with k categories is grouped into k clusters and n is the total number of documents in the data set. Given a particular category L_r of size n_r and a particular cluster S_i of size m_i, n_{ri} denotes the number of documents belonging to category L_r that are assigned to cluster S_i. Table 3.7 presents the two evaluation metrics used in this paper.

The first metric is the FScore (68), which measures the degree to which each

FScore	$\sum_{r=1}^{k} \frac{n_r}{n} \cdot \max_{1 \leq i \leq k} \frac{2 \cdot R(L_r,S_i) \cdot P(L_r,S_i))}{R(L_r,S_i)+P(L_r,S_i)}$, $R(L_r, S_i) = \frac{n_{ri}}{n_r}$, $P(L_r, S_i) = \frac{n_{ri}}{m_i}$
Entropy	$\sum_{i=1}^{k} \frac{m_i}{n} \cdot (-\frac{1}{\log k} \cdot \sum_{r=1}^{k} \frac{n_{ri}}{n_i} \cdot log\frac{n_{ri}}{m_i})$

Table 3.7: Experimental metrics

cluster contain s documents from the original category. In the FScore function, R is the recall value and P is the precision value defined for category L_r and cluster S_i. The second metric is the Entropy (64), which examines how the documents in all categories are distributed within each cluster. In general, FScore ranks the clustering quality from zero (worst) to one (best), while Entropy measures from one (worst) to zero (best). The FScore value will be one when every category has a corresponding cluster containing the same set of documents. The Entropy value will be zero when every cluster contains only documents from a single category.

3.5.4 Experimental results

Evaluations on text datasets 1-4

Our first set of experiments focused on evaluating the average quality of the clustering solutions produced by the various algorithms and the influence of text dataset characteristics, such as the number of clusters and the relatedness of clusters, on algorithms. On each small-scale structured text dataset built from 20 newsgroups, we ran the above algorithms 10 times. The average FScore and Entropy values are shown in the following bar graphs. The results of the average FScore values for the six algorithms on datasets 1-4 are shown in the left graph of Figure 3.3, while a similar comparison based on the Entropy measure is presented in the right graph.

A number of observations can be made by analyzing the results in Figure 3.3. First, TCPSO, Bisection k-means and Graph-based yield clustering solutions that are consistently better than those obtained by the other algorithms in all experiments on text datasets 1-4. This is true whether the clustering quality

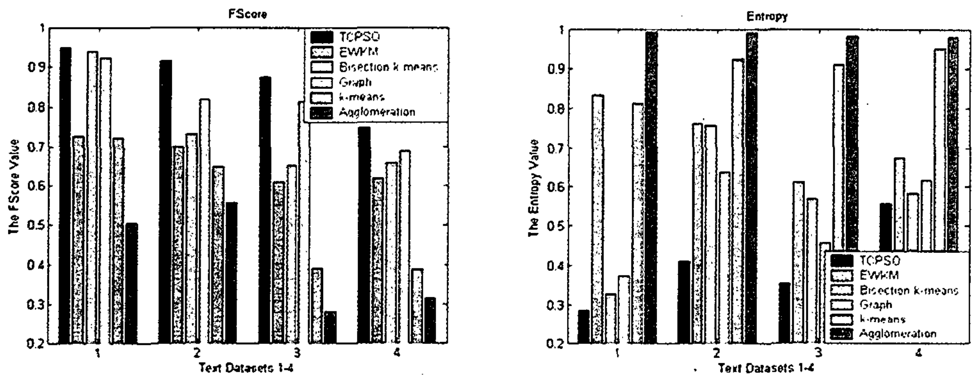


Figure 3.3: FScore and Entropy values of each algorithm on datasets 1-4.

is evaluated using the FScore or the Entropy measure. These three algorithms produced solutions that are about 5-40% better in terms of FScore and around 5-60% better in terms of Entropy.

Second, TCPSO yielded the best solutions irrespective of the number of clusters, the relatedness of clusters and the measure used to evaluate the clustering quality. Over the first set of experiments, the solutions achieved by TCPSO are always the best. On average, TCPSO outperforms the next best algorithm, Graph-based, by around 3-9% in terms of FScore and 9-21% in terms of Entropy. In general, it achieves higher FScore values and lower Entropy values than the other algorithms. The results in the FScore graph are in agreement with those in the Entropy graph. Both of them show that it greatly surpasses other algorithms for text clustering.

In general, the performances of TCPSO, Graph-based and Bisection k-means deteriorate on text datasets where categories are highly related. In such documents, each category has a subset of words and two categories share many of the same words. Basically, the probability of finding the true centroid for a document varies from one dataset to another, but it decreases as category relatedness increases. Thus, we would expect that a document can often fail to yield the right centroid regardless of the metric used.

Third, except for the Entropy measure on text datasets 1 and 4, Graph-based performs the next best for both FScore and Entropy. Bisection k-means also performs quite well when the categories in datasets are highly unrelated. EWKM always performs in the middle range, due to its employment of the local search strategy. The solutions yielded by the k-means fluctuate in all experiments. Agglomeration performs the worst of the six algorithms, yielding small FScore and large Entropy values on all datasets. This means the number of documents from each cluster yielded by Agglomeration is thus more balanced in one genuine category and as a result, the number of documents that are correctly identified by Agglomeration in the genuine category is very low.

Evaluations on text datasets 5-8

To further investigate the behavior of TCPSO, we performed a second series of experiments, focused on evaluating the comprehensive performance of each algorithm on large text datasets. Each algorithm was independently run 10 times on each of datasets 5-8. The average FScore values for the six algorithms on datasets 1-4 are shown in the left graph of Figure 3.4, while a similar comparison based on the Entropy measure is presented in the right graph.

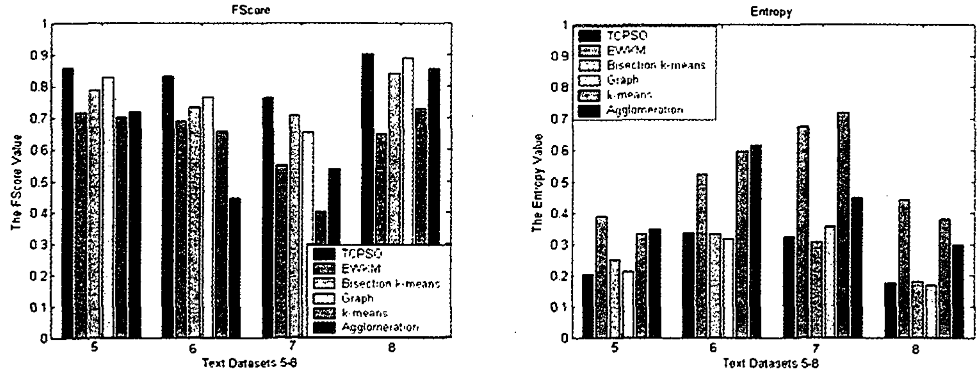


Figure 3.4: FScore and Entropy values of each algorithm on datasets 5-8.

Looking at the two graphs of Figure 3.4 and comparing the performance of each algorithm on datasets 5-8, we can see that TCPSO outperformed all other algorithms, on average, although Graph-based and Bisection k-means are very close. Graph-based and Bisection k-means are respectively worse than TCPSO in terms of FScore by 1%-11% and 6%-10%, on average, while they yield similar Entropy values to TCPSO. That means each category of documents is mainly distributed within its dominant cluster and TCPSO recovers more documents of each cluster primarily from their original categories.

Although Graph-based and Bisection k-means performed quite well on the datasets $tr41$ and $k1b$, their performances deteriorate sharply on the datasets $tr45$ and wap, where there is considerable overlap between the key words of different categories, such as the categories cable and television, multimedia and media etc. They failed to perform well on such datasets, while TCPSO still achieved relatively high FScore values because it identifies clusters embedded in subspaces by iteratively assigning an optimal feature weight vector to each cluster. The other three algorithms occasionally work well, achieving FScore values above 0.8 on the dataset $k1b$, whose categories (business, politics, sports etc.) are highly unrelated, but the FScore values are relatively low on other datasets. Moreover, their corresponding Entropy values are very high, which means each category of documents is more evenly distributed within all the clusters.

3.6 Conclusions

In response to the needs arising from the emergence of high-dimensional datasets in data mining applications and the low cluster quality of projected clustering algorithms, this paper proposed a particle swarm optimizer for the variable weighting problem, called PSOVW. The selection of the objective function and the search strategy is very important in soft projected clustering algorithms. PSOVW employs a k-means weighting function, which calculates the sum of the within-cluster distance for each cluster along relevant dimensions preferentially to irrelevant ones. It also proposes a non-normalized representation of variable weights in the objective function, which greatly facilities the search process. Finally, the new algorithm makes use of particle swarm optimization to get near-optimal variable weights for the given objective function. Although PSOVW runs more slowly than other algorithms, its deficiency in running time is acceptable. Experimental results on synthetic and real-life datasets, including an application to document clustering, show that it can greatly improve clustering accuracy for high-dimensional data and is much less sensitive to the initial cluster centroids. Moreover, the application to text clustering shows that the effectiveness of PSOVW is not confined to Euclidean distance. The reason is that the weights are only affected by the values of the objective function and weight updating in PSOVW is independent of the similarity measure. Other similarity measures can thus be used for different applications: the extended Jaccard coefficient for text data, for instance.

Two issues need to be addressed by the PSOVW algorithm. PSOVW is still not able to recover the relevant dimensions totally, similar to other soft projected clustering algorithms. The final weights obtained under the guidance of the objective function cannot totally capture the effect of the variances along each dimension. Some large weights do not adequately reflect the importance of the corresponding dimensions, although they do not have a significant negative impact on cluster quality. The other major issue in PSOVW is that usually the structure of the data is completely unknown in real-world data mining applications and it is thus very difficult to provide the number of clusters k. Although a number of different approaches for determining the number of clusters automatically have been proposed (28; 53), no reliable method exists to date, especially for high-dimensional clustering.

Chapter 4

PSO Clustering with Auto-determination of k

In this chapter, our aim is to develop another particle swarm optimizer, called autoPSO, for the problem of clustering high-dimensional data. As well as identifying clusters, the aim here is to automatically determine the correct appropriate number of clusters. The objective function employed is the Davies-Bouldin (DB) index, which is based on the ratio of within-cluster scatter to between-cluster separation. The DB index makes it possible to directly compare partitions with similar or different numbers of clusters, in the same generation or between adjacent generations.

Given the objective function, clustering is formulated as a continuous function optimization problem with bound constraints. In order to encode a variable number of clusters, autoPSO also utilizes a real-number matrix and a binary vector representation for a particle. Then, a new crossover matrix learning procedure, derived from (30) and governed by the associated binary vector, is proposed to maintain the population diversity, making autoPSO immune to the premature convergence problem. Experimental results on both synthetic high-dimensional data sets from a data generator and handcrafted low-dimensional data sets from Handl (28; 31) show that it is able to correctly identify clusters of high-dimensional data without needing to rely on a given number of clusters k.

This chapter is organized as follows. Section 1 reviews some previous related work on automatically determining the number of clusters k. Section 2 presents our enhanced PSO algorithm for clustering high-dimensional data and determining the number of clusters. Simulations on synthetic high-dimensional data produced by a generator and handcrafted low-dimensional data from Handl (31) are given in Section 3, which contains the description of data sets with different characteristics, the experimental setting for autoPSO, and the experimental results. The conclusions and directions for future work are given in Section 4.

4.1 Automatically Determination of k in Clustering

A fundamental difficulty in cluster analysis is the determination of the number of clusters. Most popular clustering algorithms require the number of clusters to be provided as an input parameter, which is difficult to set when the structure of the data is not completely known a priori. In this situation, automatically estimating the number of clusters and simultaneously finding the clusters becomes a challenge.

Due to good performance of stochastic search procedures, one way to automatically determine the number of clusters is to make use of evolutionary techniques. In this regard, genetic algorithms (GA) have been the most frequently proposed for automatically clustering data sets. Basically, two types of GA representations for clustering solutions have been explored in the literatures: partition-based and centroid-based representations. Partition-based encodings directly represent the cluster membership of the i^{th} data object by the i^{th} gene. Although this is a straightforward encoding, it does not reduce the size of the search space and makes it difficult to design effective search operators. For this reason, many researchers have chosen to use centroid-based encodings, which borrow the idea of the popular k-means algorithm. The representation is encoded as the cluster centroids (7; 8; 25; 39; 59) and each data object is subsequently assigned to the closest cluster centroid.

In (7), Bandyopadhyay and Maulik proposed a genetic algorithm in which a variable-length real-number string representation is used to encode the coordinates of the cluster centroids in the clustering problem. Chromosome i is encoded as be $m * k_i$, where m denotes the number of data features and k_i denotes the number of clusters of the i^{th} individual. New crossover and mutation operators are defined for tackling variable string lengths and real numbers. In (8), they also presented a fixed string representation composed of real numbers and the do-not care symbol '$\sharp$' to encode a variable number of clusters. These two genetic algorithms have been demonstrated to evolve the number of clusters as well as to providing a correct clustering. However, their experiments were based on data sets which tend to be low-dimensional and spherical in shape.

Tseng and Yang (59) proposed a scheme in which the data set is first split and then the smaller clusters are merged using a genetic algorithm. The proposed method has two stages, the nearest-neighbor clustering stage and a genetic clustering stage. In the first stage, small clusters are obtained, while the second one groups the small clusters into larger ones. Their GA-based clustering algorithm, called CLUSTERING, can automatically search for the correct number of clusters and it is suitable for clustering data with compact spherical clusters. It was also applied to a large data set which has 20,000 features. However, the algorithm failed to identify clusters which are confined partially or fully within another cluster.

For one general clustering problem, Lai (39) employed another algorithm based on a hierarchical genetic algorithm, in which an individual is composed of two types of genes, the control and parameter genes. The control gene is encoded in binary digits, where the number of "1s" represents the number of clusters. The parameter gene is encoded in real numbers, which represent the coordinates of the cluster centroids. The parameter gene is governed by the control gene. Where the control gene is "1", the corresponding parameter gene is activated; otherwise, the associated parameter gene is disabled.

Lin et. al. in (25) utilized a genetic algorithm with binary rather than string representation to encode a variable number of cluster centroids. The proposed method selects cluster centroids directly from the data set and speeds up the fitness evaluation by constructing a look-up table in advance and saving the distances between all pairs of data points. Suitable operators are introduced. The proposed algorithm has been demonstrated by the authors to provide stable performance in terms of number of clusters and clustering results.

In (38), Liu et al employed a genetic algorithm with the same encoding $m{*}k_i$ as (7) to represent the coordinates of k_i cluster centroids. They designed two special operators, noising selection and division-absorption mutation, for the clustering problem. The proposed method can automatically provide the number of clusters and find the clustering partition.

In (28; 31), Handl et al employed a multiobjective genetic algorithm called MOCK to solve the clustering problem. For the encoding, they employed the locus-based adjacency representation introduced in (48), where each individual consists of n genes, in which n is the size of the data set and each gene can take integer values j in the range of $[1, n]$. Thus, a value j assigned to gene i is meant as a link between two data objects i and j. In the result, they will be assigned to the same cluster. MOCK optimizes two complementary clustering objectives, within-cluster compactness and connectivity, and attempts to automatically estimate the number of clusters in the data set. MOCK has achieved high quality clustering results on the authors' data sets, which have complex cluster shapes, such as spherical, ellipsoidal, or long datasets and overlapped clusters. However, it is not readily applicable on high-dimensional datasets where clusters are embedded in subspaces.

4.2 PSO Clustering with Auto-determination of k

In this section, we propose a particle swarm optimizer, called autoPSO, to automatically determine the number of clusters k and simultaneously identify clusters.

4.2.1 Encoding

To apply PSO to the clustering problem, one needs to choose an objective function and a suitable encoding of a partition.

For the encoding, the position of each particle consists of two components, a real-number matrix $M_{k_{max}*m}$ and a binary column vector $Con_{k_{max}}$. Here, k_{max} is the maximal number of clusters, which can be set by the user and m is the number of dimensions. Each row in the matrix is coded as real numbers representing the coordinates of a cluster centroid. A "1" in the binary vector signifies that there is a row of cluster centroid in the corresponding row of the matrix, while "0" signifies there is no center centroid in the corresponding row. Thus, the matrix is "supervised" by the vector. Where the binary vector value is "1", the associated row of the matrix is activated; otherwise the associated row is disabled.

In the autoPSO algorithm, we need to initialize three swarms.

- In the position swarm, a position is a real-number matrix M. For a position i in the population, a random integer k in the range of $[2, k_{max}]$ is generated. Then, k different data objects are randomly chosen from the data set as k initial center centroids. These data objects are randomly distributed in rows of the matrix.

  For example, assume that $D = 4$, $k_{max} = 6$, and random number $k_i = 4$ for particle i. Then 4 data objects are randomly chosen as 4 cluster centroids. Assume that these 4 cluster centroids are randomly distributed in rows $1, 5, 3, 6$ of the position matrix, respectively.

Let 4 data objects selected from the data set be

$$(2.4 \quad 4.3 \quad 5 \quad 0.1), (10.7 \quad 8.2 \quad 3.1 \quad 2.9),$$

$$(2 \quad 1.9 \quad 8.2 \quad 4.4), (12.9 \quad 13.1 \quad 5.3 \quad 2.8)$$

Then, the position matrix M_i for particle i will look like this:

$$\begin{array}{cccc} 2.4 & 4.3 & 5 & 0.1 \\ & * & * & * \\ 2 & 1.9 & 8.2 & 4.4 \\ & * & * & * \\ 10.7 & 8.2 & 3.1 & 2.9 \\ 12.9 & 13.1 & 5.3 & 2.8 \end{array}$$

- In the velocity swarm, a velocity is also a $k_{max} * D$ matrix, which is set to random numbers uniformly distributed in the range $[-V_{max}, V_{max}]$. Here, V_{max} means the maximum flying velocity of particles and $V_{max} = 0.25 * (x_u - x_l)$, where x_u and x_l are the upper bound and the lower bound of the search space, respectively.

- In the control parameter swarm, a control parameter is binary vector Con. In each Con_i, if the i^{th} element is 1, it indicates the presence of the associated cluster in the position matrix M_i. If the corresponding element is 0, it denotes the absence of the cluster.

  The binary vector for the above example is:

$$\begin{array}{c} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \end{array}$$

This encoding scheme has several major advantages for our application. Most importantly, there is no need to set the number of clusters in advance. Hence, we can evolve and compare solutions with different numbers of clusters in one generation or adjacent generations of PSO. Furthermore, this real-number representation is well suited for the use of arithmetic operators in the original PSO and a complex crossover operator which will be introduced later into PSO.

4.2.2 The Objective Function

The Davies-Bouldin (DB) index (13) is a popular internal validation index which captures both within-cluster scatter and between-cluster separation in a nonlinear combination. It is defined as:

$$DB(z) = \frac{1}{k(z)} \cdot \sum_{i=1}^{k} R_i(z) \tag{4.1}$$

where

$$R_i(z) = \max_{i,j \neq i} \frac{S_i(z) + S_j(z)}{B_{ij}(z)}$$

is the ratio of the sum of within-cluster scatter

$$S_i(z) = \frac{1}{|C_i(z)|} \sum_{x \in C_i} d(x, z_i) \tag{4.2}$$

to between-cluster separation

$$B_{ij}(z) = d(z_i, z_j) \tag{4.3}$$

Here, x is the data object and Z_i denotes the cluster centroid of cluster C_i, usually defined as $z_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$, where $|C_i|$ is the number of data objects belonging to cluster C_i. d is the distance function which measures the dissimilarity between two data objects. DB takes values in the interval $[0, \infty)$ and needs to be minimized.

The use of the DB index has some major advantages. First, it is unbiased with respect to the number of clusters. It can measure the quality of partitions with similar or different numbers of clusters. Hence, we can compare solutions with different numbers of clusters. Second, values returned by the DB index are easier to interpret. The smaller the value of the DB index, the more compactness in individual clusters and the more separation between clusters.

Given a previous set of cluster centroids, the cluster membership of each data object is calculated by the following formula.

Let $q(t) = \sum_{j=1}^{m} [(x_{i,j} - z_{t,j})^2]$, $q^* = \min q(t)$, $1 \leq t \leq k$

Then,

$$u_{l,i} = \begin{cases} 1, & \text{if } q(t) = q^* \\ 0, & \text{otherwise} \end{cases} \tag{4.4}$$

Once the cluster membership is obtained, the cluster centroids are updated by

$$z_{l,j} = \frac{\sum_{i=1}^{n} u_{l,i} * x_{i,j}}{\sum_{i=1}^{n} u_{l,j}}, for \quad 1 \leq l \leq k \quad and \quad 1 \leq j \leq m \tag{4.5}$$

In our implementation, if membership update by formula 4.4 results in an empty cluster, we randomly select a data object from the data set to reinitialize the centroid of the cluster. Then we can calculate the fitness values, namely the values of the DB index, for the positions in the swarm.

4.2.3 Operators

In order to maintain population diversity and prevent premature convergence, autoPSO borrows the idea of using *CpBest* instead of *gBest* and *pBests* from (30). The use of *gBest* causes the original PSO to converge prematurely to local minima. In autoPSO, the position swarm and the control parameter swarm both need the crossover operation, which governs the behavior of the position swarm. The velocity update is defined as:

$$V_i^d(t+1) = w * V_i^d(t) + c1 * r1 * (CpBest_i^d - X_i^d(t)) \tag{4.6}$$

where *CpBest* rather than *gBest* guides particles to move towards better regions. Each component of *CpBest* is obtained by a crossover operation between the individual's own *pBest* and one of the other best personal positions in the swarm, represented here by s_{pBest}.

The tournament mechanism is employed to select s_{pBest}, with the consideration that a particle learns from a good exemplar. First, two individuals, $pBest_1$

and $pBest_2$, are randomly selected. Then their objective function values are compared: $s_{pBest} = pBest_1$ if $F(pBest_1) < F(pBest_2)$, $s_{pBest} = pBest_2$ otherwise. The crossover is done as follows: Each element at a position of $CpBest$ is assigned the value either from s_{pBest} or from $pBest$ at the corresponding position. This assignment is made randomly according to a user-defined probability value P_c. If, despite the random assignment process, all the elements of $CpBest$ take values from the position's own $pBest$, one element of $CpBest$ will be randomly selected and its value will be replaced by the value of the corresponding position from s_{pBest}.

The arithmetic operators provided by the original PSO are very effective for continuous function optimization. However, the arithmetic operators, such as the "-" operator between $CpBest$ and X, needs special treatment in autoPSO because of the incorporation of the control parameter swarm. There are two values in control parameter vectors, namely "0" and "1". There are four combinations of the control parameter values between $CpBest$ and X, as follows:

The control parameter of	$CpBest$	-	X	=	result
	0		0	=	0
	0		1	=	0
	1		0	=	1
	1		1	=	1

Consequently, the result has the same control parameter vector as $CpBest$. However, the number of "1s" in the control parameter vector of the result sometimes happens to be less than 2 after the crossover operator, i.e., there are less than two cluster centroids in the corresponding result. In this situation, we keep the control parameter of X as the result, in order to guarantee that there exist at least two clusters in the data set.

4.2.4 Pseudo Codes for AutoPSO

The autoPSO algorithm is summarized as follows:

```
Initialization: Set the maximal number of clusters, $k_{max} = 10$.
                Set the parameters in PSO, namely $w$, $c1$, $P_c$.
                For each particle $i$,

                    Randomly choose $k_i$ data objects from the data;
                    set as the initial cluster centroids;
                    Place these cluster centroids in $k_i$ random rows
                    to construct the position matrix $pos$;
                    Construct the velocity matrix $vel$, randomly uniformly
                    distributed in the range $[-mv, mv]$, where
                    $mv = 0.25 * (x_u - x_l)$, and $x_u$, $x_l$ are the
                    upper and lower bounds of the dimension space.
                    Evaluate the fitness value for each position
                    matrix by 4.1.
                    Generate $CpBest$ and its associated control parameter.
                    Vector $Con$ for each position matrix.

Repeat:         For each particle $i$,

                    If the fitness value of a position matrix isn't improved within
                    5 iterations,
                        Update its associated $CpBest$ and $Con$.
                    Update the velocity and position matrix, $pos$ and $vel$.
                    Confine $vel$ in the range of $[-mv, mv]$.
                    Confine $pos$ in the range of $[x_l, x_u]$.
                    Evaluate the fitness value of each position.
                    Update $pBest, gBest$, and $Con$.

Until           Find the global fitness value 0,
                or reach the maximal number of fitness evaluations.
```

4.3 Experimental Results

4.3.1 Contestant Algorithms

In this section, we study the performance of autoPSO by comparing the algorithm to:

1. three traditional clustering algorithms: k-means, Bisection k-means, and Graph-based clustering.

2. PSOVW

3. GA-based clustering (8)

The four clustering algorithms in the first two points require the number of clusters, k, to be provided by the user, while GA-based clustering allows us to automatically estimate the correct number of clusters as well as finding the clusters. The k-means is a conceptually simple and well-proven algorithm. The reason we have selected the Bisection k-means and Graph-based clustering algorithms for comparison is that both of them perform well on high-dimensional data sets, such as text data, as shown in Chapter 3. PSOVW is an effective soft projected algorithm for clustering high-dimensional data based on variable weighting, already introduced in Chapter 3. Finally, the choice of GA-based clustering (8) reflects our wish to demonstrate that autoPSO achieves a high level of performance not because it employs new objective functions, but rather because autoPSO introduces a more effective search strategy.

k-means

The k-means algorithm starts with a random partitioning and loops the following steps: 1) Computes current cluster centroids and the average vector of each cluster in a data set in some cases; 2) Assign each data object to the cluster whose cluster centroid is closest to it. The algorithm terminates when there is no further change in the cluster centroids. By this means, k-means locally maximizes compactness by minimizing the within-cluster scatter, namely the sum of squares of the differences between data objects and their corresponding cluster centroids. In our implementations, an empty cluster sometimes occurs, so we reassign one data object randomly selected from the data set to it.

Bisection k-means

In order to generate k desired clustering solutions, Bisection k-means performs a series of k-means. Bisection k-means is initiated with the universal cluster containing all data objects. Then it loops: it selects the cluster with the largest variance and calls k-means, which optimizes a particular clustering criterion function in order to split this cluster into exactly two subclusters. The loop is repeated a certain number of times such that k non-overlapping clusters are generated. Note that this approach ensures that the criterion function is locally optimized within each bisection, but in general it is not globally optimized.

Graph-based

In the Graph-based clustering algorithm, the desired k clustering solutions are computed by first modeling the objects using a nearest-neighbor graph. Each data object becomes a vertex, and each data object is connected to the other objects most similar to it. Then, the graph is split into k clusters using a min-cut graph partitioning algorithm.

PSOVW

See Chapter 3.

GA-based

Finally, we compared against the genetic algorithm proposed by Bandyopadhyay (8). It starts with a randomly initialized population with fixed-length strings representing the centers of a number of clusters, whose value may vary. The genetic clustering algorithm employs a conventional proportional selection, a modified single-point crossover and a special mutation. A cluster validity index such as the Davies-Bouldin (DB) index is utilized for computing the fitness of chromosomes. Experiments on four data sets, three artificial and one real-life, have demonstrated that GA-based clustering can automatically estimate the appropriate clustering of a data set.

4.3.2 Data Sets

The clustering performance of the autoPSO algorithm was evaluated on the following two groups of synthetic data sets:

1. **High-dimensional data:**

High-dimensional data sets contain clusters embedded in different dimensional subspaces and have often been used to investigate algorihtms' performance in clustering high-dimensional data. The number of clusters and the size of the individual clusters are manually fixed, while the mean and the standard deviation vectors are randomly generated in a certain range. The method for producing these high-dimensional data sets is given in Chapter 3, which provides a detailed description of the data sets. These data sets consist of 4 or 10 clusters, combined with different dimensions, such as 20, 100, 1000 and 2000 dimensions. We selected a total of 8 type 2 synthetic high-dimensional data sets, from Chapter 3.

2. **Complex 2-dimensional data:**

   We selected 6 2-dimensional data sets with square, long, smile and spiral shapes from (31). These 2-dimensional data sets were used to analyze the algorithms'

   performance with respect to specific data characteristics, such as cluster shape. The data sets, except for the smile data set, are described by 2-dimensional normal distributions $N(\mu_i, \sigma_i), i = 1, 2$. The number of clusters, the sizes of individual clusters, the mean and the standard variance for each normally distributed set are manually fixed.

   The 2-dimensional data are summarized in Table 4.1. m is the number of dimensions, k denotes the number of clusters, and n_i gives the number of data objects for cluster i. The test sets are generated by either Normal or Uniform distributions, i.e., $N(\mu, \sigma)$ or $U(\mu, \sigma)$, where μ and σ are the mean and the standard deviation, respectively. For the Smile data set, circles $C(u, r, \mathit{start} \sim \mathit{end})$ are additionally used, where u is the center of the circle, r is its radius, and $\mathit{start} \sim \mathit{end}$ describes the part of the circle that is actually drawn.

Datasets	k	n_i	m	Normal Distribution
Square1	4	4*125	2	N([0, 0] [2, 2]), N([10, 10], [2, 2]), N([0, 10], [2, 2]), N([10, 0], [2, 2])
Square2	4	4*125	2	N([0, 0] [2, 2]), N([6, 6], [2, 2]), N([0, 6], [2, 2]), N([6, 0], [2, 2])
Square3	4	200, 100 100, 100	2	N([0, 0] [2, 2]), N([10, 10], [2, 2]), N([0, 10], [2, 2]), N([10, 0], [2, 2])
Long	2	2*250	2	N([0.3, 0.7], [0.01, 0.01]), N([0.7, 0.7], [0.01, 0.01])
Smile	4	4*125	2	N([0, 0], [1, 0.1]), N([0, 1], [1, 0.1]), C([0.5, 0.5], 0.5, $0 \sim 2\pi$), C([0.5, 0.5], 0.3, $1.25\pi \sim 1.75\pi$), +U([0, 0], [0.1, 0.1])
Spiral	2	2*250	2	

Table 4.1: 2-dimensional datasets.

Figure 4.1 gives a visual depiction of the 2-dimensional synthetic data sets. In *Square*1, *Square*2 and *Square*3, the degree of overlap and the size of individual clusters are varied. *Long* contains elongated cluster shapes. Three clusters in *Smile* are contained within another one, while *Spiral* consists of interlaced clusters.

The synthetic data requires no processing and the distance computation is done using the Euclidean distance.

4.3.3 Experimental Metric

The clustering performances of all of the above six algorithms, including autoPSO, were compared using the *FScore* evaluation measure. Basically, this measure is based on the ideas of precision and recall from information retrieval. This function is an external function that compares the clustering solutions to the original class labels. Assume that a data set with k categories is grouped into k clusters and n is the total number of documents in the data set. Given a particular category L_r of size n_r and a particular cluster S_i of size m_i, n_{ri} denotes the number of documents belonging to category L_r that are assigned to cluster S_i. The $FScore$ measure is defined as follows:

$$FScore = \sum_{r=1}^{k} \frac{n_r}{n} \cdot \max_{1 \le i \le k} \frac{2 \cdot R(L_r,S_i) \cdot P(L_r,S_i))}{R(L_r,S_i)+P(L_r,S_i)}$$

$$R(L_r, S_i) = \frac{n_{ri}}{n_r} \tag{4.7}$$

$$P(L_r, S_i) = \frac{n_{ri}}{m_i}$$

where P and R are the precision and recall for each class i and cluster j. $FScore$ is limited to the interval [0,1] and should be maximized for the optimal clustering.

4.3.4 Experimental Results

Since k-means, Bisection k-means, PSOVW, GA-based and autoPSO are randomly initialized algorithms, we performed 20 runs of each algorithm on each data set and averaged the results obtained. We compare the different approaches in terms of the measure described in formula 4.7. For k-means, Bisection k-means, Graph-based and PSOVW, we set the proper number of clusters for each dataset. GA-based and autoPSO do not require advance setting of this parameter.

Our first set of experiments is focused on evaluating the ability of each algorithm to cluster high-dimensional data with clusters embedded in different dimensional subspaces, namely with clusters having different relevant dimensions. The average quality of the clustering solutions produced by the various algorithms is reported in Table 4.2. Figure 4.2 illustrates the distribution of the $FScore$ values in Table 4.2.

In table 4.2, the reported results for PSOVW do not correspond with the results presented in Chapter 3. The reason is that different experimental measures are used in these two series of experiments. In Chapter 3, we employ the precision measure, while we use the FScore measure, which is a combination of precision and recall, in Chapter 4. Actually, FScore is more appropriate to measure the performances of clustering algorithms than precision used in Chapter 3. However, the performances of LAC, EWKM, W-k-means and k-means work badly on high-dimensional datasets, it is therefore very difficult to calculate the FScore values for their results.

High-dimensional Data Sets	autoPSO	GA-based	PSOVW	k-means	Bisection k-means	Graph based
$4k20m$	**100**	**100**	**100**	64.9	100	88.7
$4k100m$	**100**	**100**	**100**	62.1	100	85.6
$4k1000m$	**100**	**100**	**100**	76.3	100	78.5
$4k2000m$	**100**	**100**	**100**	72.4	100	87.3
$10k20m$	**100**	90.5	93.2	35.1	92	86.4
$10k100m$	**100**	87.8	92.4	33.4	92.5	85.5
$10k1000m$	**100**	91.3	92.6	31.9	87	88.9
$10k2000m$	**100**	88.9	94.8	36.2	86.2	81

Table 4.2: The *FScore* values of each algorithm on eight high-dimensional datasets.

From Table 4.2 and Figure 4.2, we note that autoPSO, GA-based, PSOVW and Bisection k-means attain comparable clustering results and all of them achieve 100 percent clustering accuracy on each trial over the data sets with 4 well-separated clusters. In our experiments on the data sets with 10 overlapped clusters, autoPSO performs most robustly compared to its contestants. It reliably generates solutions that are comparable to or better than those of the other algorithms, showing that it explores high-quality clustering results on high-dimensional data sets. Evidently, it manages to cope with the complicated data sets where the relevant dimensions of individual clusters overlap considerably, while all of the other clustering methods fail on certain of these data sets.

As can be expected, k-means performs very poorly in the absence of overlapped clusters, as in the data sets with 10 clusters. The clustering performance of the Graph-based algorithm is little affected by the number of dimensions, because it employs the graph idea to find clusters. Bisection k-means produces a hierarchy by random binary splits, resulting in higher variance.

On all of the high-dimensional data sets, autoPSO reliably identifies the correct number of clusters. One should note that the obvious difficulty of determining the number of clusters in high-dimensional data sets. In our experiments, the GA-based clustering algorithm is able to find the correct number of clusters as well as identifying clusters on the data sets with 4 clusters. However, its performance is inferior to that of autoPSO on the data sets with 10 clusters. The good performance of autoPSO in clustering high-dimensional data does not result from the objective function which is not new, but from its more effective search strategy.

To further investigate the behavior of autoPSO on low-dimensional data sets, we performed a second series of experiments, focusing on evaluating the performance of each algorithm on 2-dimensional data sets with different cluster shapes. Each algorithm was independently run 10 times on each 2-dimensional data set. The average FScore value for each algorithm is shown in Figure 4.3.

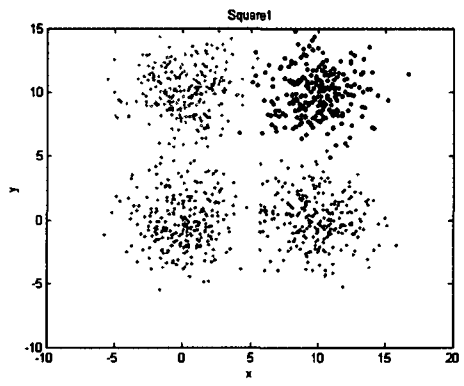
Figure 4.3 shows the results in terms of $FScore$ over 2-dimensional data sets for all of the clustering approaches. As can be seen from Figure 4.3, in our experiments, autoPSO and GA-based do not work well on the data sets with overlapped and interlaced clusters, *Square*2 and *Spiral*, because they optimize compactness and spatial separation, not connectedness. PSOVW achieves poor performances on all the 2-dimensional data sets except for *Long*. The main reason is that PSOVW identifies clusters by variable weighting. Usually, it is able to get to a set of variable weights, which make the data set separable. However, 2-dimensional data sets are usually inseparable, so it cannot perform well on the 2-dimensional data sets with inseparable clusters.

k-means performs well on the data sets *Square*1, *Square*2 and *Square*3, as they contain spherical clusters, while it fails for the data sets with *Long*, *Smile* or *Spiral* shapes. The main reason for this is that it optimizes compactness, not connectedness or spatial separation. Bisection k-means leads to better clusters than those obtained by k-means on most of runs. Graph-based can detect clusters of any arbitrary, and it achieves about the same performances on all the 2-dimensional data sets. However, it tends to show a poor performance on overlapped clusters.

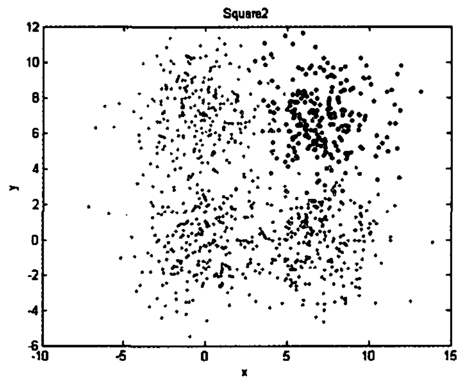
4.4 Conclusions

AutoPSO has shown very good performance in terms of providing the correct number of clusters and identifying clusters over high-dimensional data sets with clusters embedded in different dimensional subsets. AutoPSO is a k-means type of clustering algorithm. It aims to optimize a set of cluster centroids, utilizing a cluster validity index, the DB index, as the objective function to be optimized. In autoPSO, each individual is encoded as a position matrix and a control vector which governs the associated matrix. Finally, it utilizes an advanced particle swarm optimizer instead of local search strategies to optimize the given objective function. The encoding makes it easier for autoPSO to search in the dimension space. Compared with the other clustering algorithms considered in this chapter, the more effective search strategy explains the superior performance of autoPSO.

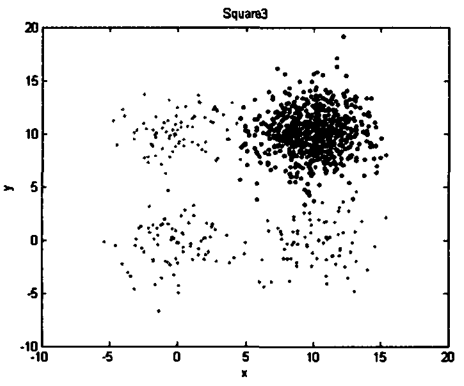
Despite its promising performance on high-dimensional data sets, autoPSO has some inherent limitations. Due to the objective scheme which optimizes compactness and separation, not connectivity, autoPSO can not differentiate heavily overlapped and interlaced clusters, such as those in $Square2$. Therefore, a more effective objective function that grasps the definition of clustering should be proposed. To sum up, although autoPSO is not simply the best optimizer, it can be a useful alternative for providing the number of clusters and identifying clusters in high-dimensional data sets.



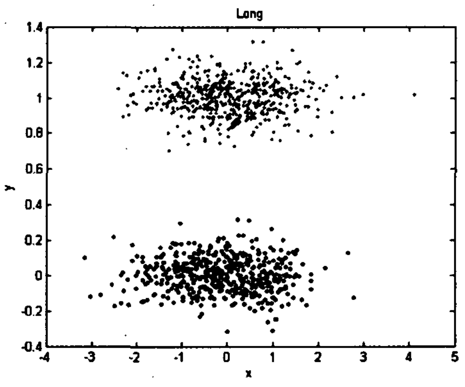
(a) Square 1



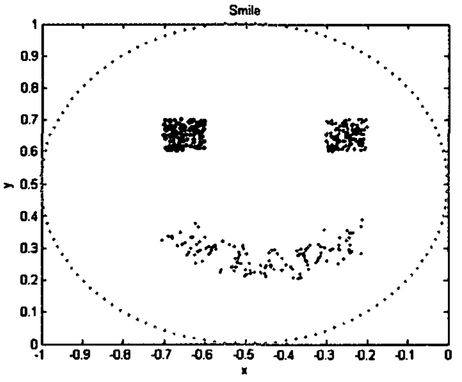
(b) Square 2



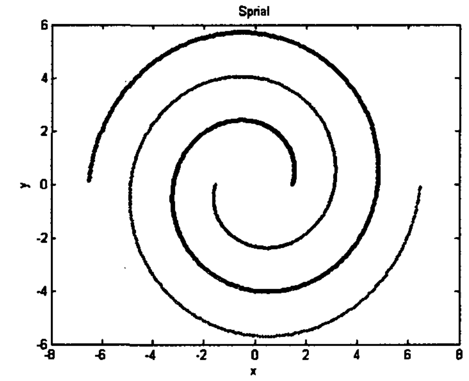
(c) Square 3



(d) Long



(e) Smile



(f) Spiral

Figure 4.1: Visual depiction of the 2-dimensional synthetic data sets.

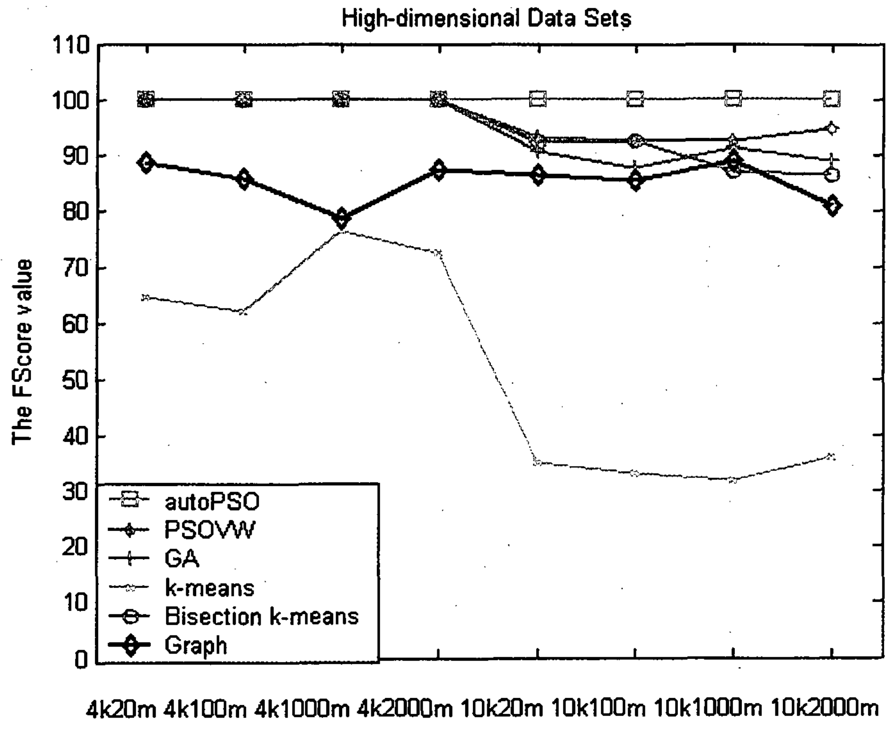


Figure 4.2: The *FScore* on high-dimensional datasets.

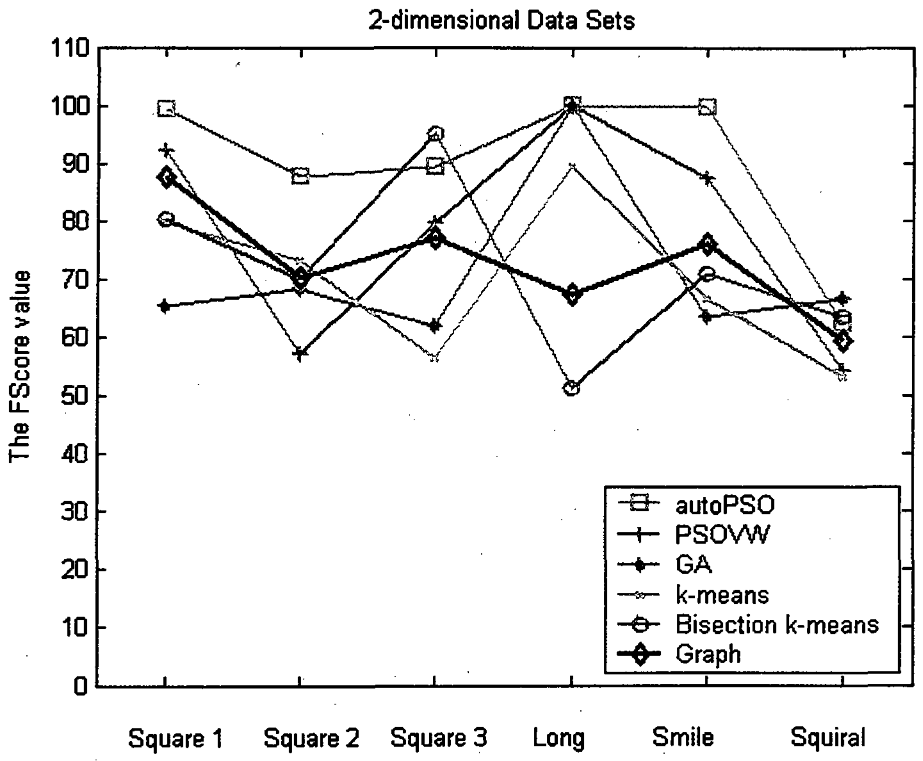
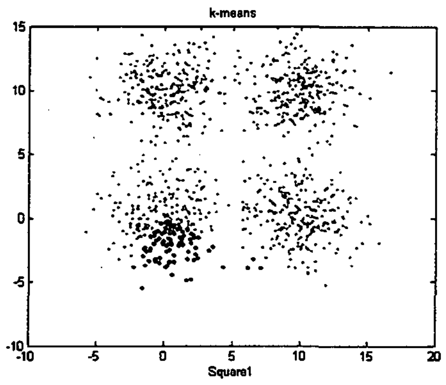
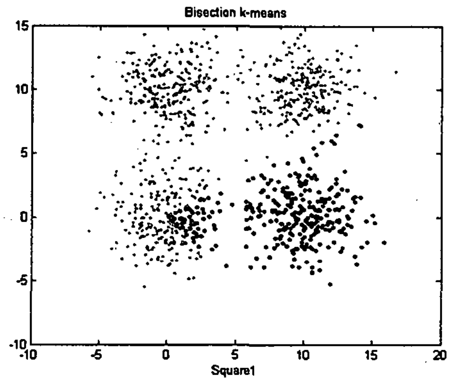


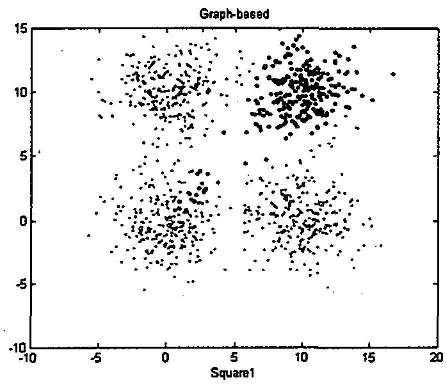
Figure 4.3: The *FScore* on 2-dimensional datasets.



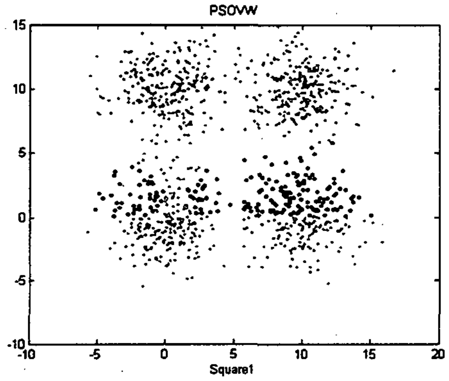
(a)k-means on $Square1$



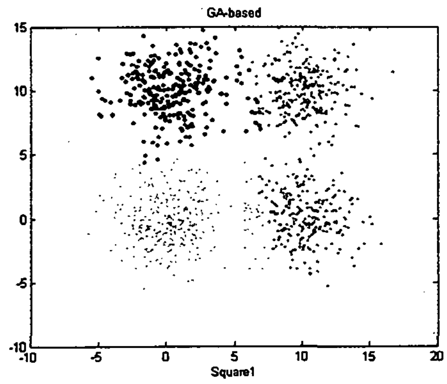
(b)Bisection k-means on $Square1$



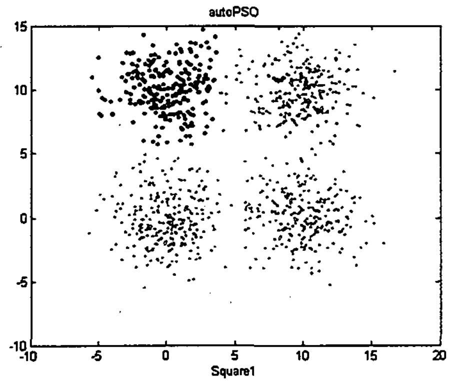
(c)Graph-based on $Square1$



(d)PSOVW on $Square1$



(e)GA-based on $Square1$



(f)autoPSO on $Square1$

Figure 4.4: Some experimental results of each algorithm on the data set, $Square1$.

Chapter 5

Conclusion and Future Work

This thesis has explored the advantages of particle swarm optimization in the context of clustering high-dimensional data. Particle swarm optimization techniques for two specific tasks have been developed and their performance has been analyzed in comparison to that of existing clustering approaches and several related projected clustering algorithms. The analysis has shown that particle swarm optimization can overcome some of the fundamental limitations of local search methods and may yield significant performance gains. On the other hand, the proposed particle swarm optimizers are computationally more expensive than standard approaches such as k-means-type of clustering algorithms.

Despite their promising performances, PSOVW and autoPSO have some limitations. Due to its heuristic nature of the algorithm, PSOVW is not guaranteed to obtain the most optimal weight vectors where large ones reflect the importance of the corresponding dimensions. Therefore, PSOVW is unable to recover the relevant dimensions totally, although the final weights it reaches do not have a significant negative impact on cluster quality. More effective methods for the selection of feature subsets for each cluster should thus be considered. It should also be noted that autoPSO fails to perform well on low-dimensional data sets, due to its use of the simple objective function, the DB index. Therefore, more effective objective schemes should be proposed for clustering low-dimensional data. In addition, these two approaches are very computationally expensive for some applications with heavy time or memory constraints.

To sum up, the research discussed in this thesis opens up a number of directions for future work. First, there is room to improve on the objective functions described in this thesis. The objective functions employed in our PSO algorithms are appropriate for clustering high-dimensional data where clusters are embedded in subspaces, due to their sparsity and separation. However, low-dimensional data sets such as 2-dimensional data sets are usually inseparable. This calls for the design of special objective schemes that will enable a more effective representation of clustering, in particular in the context of the problem of automatically determining the number of clusters k.

Second, there is also definitely room to use more efficient global optimization algorithms. In this thesis, global optimization methods are used as basic algorithms. The advanced particle swarm optimization algorithms PSOVW and autoPSO have been applied to high-dimensional data throughout this thesis. However, there is no reason to assume that our PSOs are superior to other variants of meta-heuristics. A comparison of implementations based on different clustering validity measures and different optimization techniques is therefore worth investigating.

Finally, since particle swarm optimization is an effective global search technique, it would be interesting to develop particle swarm approaches to unsupervised feature selection and feature subspace selection for subspace clustering.

Bibliography

[1] *http://glaros.dtc.umn.edu/gkhome/cluto/cluto/download.* 57

[2] *http://kdd.ics.uci.edu/.* 54, 57

[3] *http://people.csail.mit.edu/jrennie/20Newsgroups/.* 57

[4] *http://trec.nist.gov/.* 58

[5] A. SALMAN, I. AHMAD, S. A.-M. Particle swarm optimization for task assignment problem. *Microprocessors and Microsystems 26*, 8 (2003), 363–371. 11, 39

[6] ANGELINE, P. J. Using selection to improve particle swarm optimization. In *CEC* (1998), pp. 84–89. 10

[7] BANDYOPADHYAY, S., AND MAULIK, U. Nonparametric genetic clustering: comparison of validity indices. *IEEE Trans. on Systems, Man, and Cybernetics, Part C: Applications and Reviews 31*, 1 (2001), 120–125. 3, 67, 68

[8] BANDYOPADHYAY, S., AND MAULIK, U. Genetic clustering for automatic evolution of clusters and application to image classification. *Pattern Recognition 35*, 6 (2002), 1197–1208. 3, 67, 75, 76

[9] BERGH F, E. A. A cooperative approach to particle swarm optimization. *IEEE Trans. on Evolutionary Computation 8*, 3 (2004), 225–239. 23

[10] BLACKWELL, T. M., AND BENTLEY, P. J. Dont push me! collision-avoiding swarms. In *CEC* (2002), pp. 1691–1696. 10

[11] BOLEY, D., AND GINI, M. Document categorization and query generation on the world wild web using webace. *AI Review 11*, 3 (1999), 365–391. 58

[12] CM. PROCOPIUC, M. JONES, P. A.-T. M. A monte carlo algorithm for fast projective clustering. In *Proc. of ACM SIGMOD Int. Conf. on Management of data* (2002), pp. 418–427. 2

[13] DAVIES, D. L., AND BOULDIN, D. W. A cluster separation measure. *IEEE Trans. Pattern Analysis and Machine Intelligence 1*, 4 (1979), 224–227. 71

[14] DOMENICONI, C., AND GUNOPULOS, D. Locally adaptive metrics for clustering high dimensional data. *Data Mining and Knowledge Discovery Journal 14* (2007), 63-97. 3, 31, 32, 33, 34, 44, 46

[15] DOMENICONI, C., AND PAPADOPOULOS, D. Subspace clustering of high dimensional data. In *Proc. of SIAM International Conference on Data Mining* (2004). 3, 31, 32, 33

[16] EBERHART, R., AND KENNEDY, J. A new optimizer using particle swarm theory. In *Proc. 6th Int. Symposium on Micromachine and Human Science* (1995), pp. 39–43. 8, 9, 22

[17] E.H. HAN, D. BOLEY, E. A. Webace: A web agent for document categorization and exploration. In *Proc. of 2nd Int. Conf. on Autonomous Agents* (1998). 58

[18] ELKE ACHTERT, CHRISTIAN BOHM, E. A. Detection and visualization of subspace cluster hierarchies. *LNCS 4443* (2008), 152–163. 2

[19] EVETT, I. W., AND SPIEHLER, E. J. Rule induction in forensic science. In *KBS in Goverment* (1987), pp. 107–118. 54

[20] FAN, H. Y., AND SHI, Y. Study on vmax of particle swarm optimization. In *Proc. Workshop Particle Swarm Optimization* (2001). 9

[21] G. CHONGHUI, T. H. Global convergence properties of evolution strategies. *Mathematica Numerica Sinica 23*, 1 (2001), 105–110. 18

[22] G. MOISE, J. S. Finding non-redundant, statistically significant regions in high dimensional data: a novel approach to projected and subspace clustering. In *Proc. of the 14th ACM SIGKDD Int. Conf. on Knowledge discovery and data mining* (2008), pp. 533–541. 2

[23] G. MOISE, J. S., AND ESTER, M. Robust projected clustering. *Knowledge and Information Systems 14*, 3 (2008), 273–298. 2

[24] G.C. ONWUBOLU, M. C. Optimal operating path for automated drilling operations by a new heuristic approach using particle swarm optimization. *International Journal of Production Research 42*, 3 (2004), 473–491. 11, 39

[25] H. J. LIN, F. W. Y., AND KAO, Y. T. An efficient ga-based clustering technique. *Tamkang Journal of Science and Engineering 8*, 2 (2005), 113–122. 67, 68

[26] H. RAN, W. Y. An improved particle swarm optimization based on self-adaptive escape velocity. *Journal of Software 16*, 12 (2005), 2036–2044. 18

[27] H. YOSHIDA, K. KAWATA, Y. F.-Y. N. A particle swarm optimization for reactive power and voltage control considering voltage security assessment. *IEEE Transactions on Power Systems 15*, 4 (2000), 1232–1239. 11, 39

[28] HANDL, J., AND KNOWLES, J. Multiobjective clustering with automatic determination of the number of clusters. Tech. rep., UMIST, 2004. 64, 65, 68

[29] Hu, X., and Eberhart, R. C. Multiobjective optimization using dynamic neighborhood particle swarm optimization. In *CEC* (2002), pp. 1677–1681. 10

[30] J.J. Liang, A.K. Qin, P. S.-S. B. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Trans. on Evol. Comp. 10*, 3 (2006), 281–295. 10, 22, 23, 26, 43, 46, 65, 72

[31] Julia Handl, J. D. K. An evolutionary approach to multiobjective clustering. *IEEE Trans. Evolutionary Computation 11*, 1 (2007), 56–76. 3, 65, 66, 68, 77

[32] J.Z. Huang, Michael K. Ng, H. R., and Li, Z. Automated dimension weighting in k-means type clustering. *IEEE Trans. on Pattern Analysis and Machine Intelligence 27*, 5 (2005), 1–12. 3, 32, 35, 44

[33] Kennedy, J. Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance. In *CEC* (1999), p. 1931C1938. 10

[34] Kennedy, J., and Mendes, R. Population structure and particle swarm performance. In *CEC* (2002), p. 1671C1676. 22

[35] Kyoung-Gu Woo, Jeong-Hoon Lee, M.-H. K., and Lee, Y.-J. Findit: A fast and intelligent subspace clustering algorithm using dimension voting. *Information and Software Technology 46*, 4 (2004), 225–271. 2

[36] L. Jing, M.K. Ng, J. H. An entropy weighting k-means algorithm for subspace clustering of high-dimensional sparse data. *IEEE Trans. on Knowledge and Data Engineering 19*, 8 (2007), 1026–1041. 3, 31, 32, 34, 36, 44, 45

[37] L. Parsons, E. Haque, H. L. Subspace clustering for high dimensional data: A review. *SIGKDD Explorations Newsletter 6* (2008), 90–105. 2

[38] L. Yongguo, Y. Mao, P. J., and Hong, W. Finding the optimal number of clusters using genetic algorithms. In *2008 IEEE Conf. on Cybernetics and Intelligent Systems* (2008), pp. 1325–1330. 68

[39] Lai, C. C. A novel clustering approach using hierarchical genetic algorithms. *Intelligent Automation and Soft Computing 11*, 3 (2005), 143–153. 3, 67

[40] Lovbjerg, M., and Krink, T. Extending particle swarm optimizers with self-organized criticality. In *CEC* (2002), p. 1588C1593. 10

[41] M. Clerc, J. K. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation 6*, 1 (2002), 58–73. 9

[42] M. Lovbjerg, T. K. R., and Krink, T. Hybrid particle swarm optimizer with breeding and subpopulations. In *GECCO* (2001), p. 469C476. 10

[43] Makarenkov, V., and Legendre, P. Optimal variable weighting for ultrametric and additive trees and k-means partitioning: Methods and software. *Journal of Classification 18*, 2 (2001), 245–271. 32

[44] Mangasarian, O., and Wolberg, W. Breast cancer diagnosis via linear programming. *SIAM News, 23*, 5 (1990), 1–18. 54

[45] M.F. Tasgetiren, M. Sevkli, Y.-C. L.-G. G. Particle swarm optimization algorithm for single machine total weighted tardiness problem. In *CEC* (2004), pp. 1412–1419. 11, 39

[46] Miranda, V., and Fonseca, N. New evolutionary particle swarm algorithm (epso) applied to voltage/var control. In *Proc. 14th Power Syst. Comput. Conf.* (2002), p. http://www.pscc02.org/papers/s21pos.pdf. 10

[47] P. Kumsawat, K. A., and Srikaew, A. A new approach for optimization in image watermarking by using genetic algorithms. *IEEE Trans. on Signal Processing 53*, 12 (2005), 4707–4719. 3

[48] Park, Y., and Song, M. A genetic algorithm for clustering problems. pp. 568–575. 68

[49] Parsopoulos, K. E., and Vrahatis, M. N. Upsoa unified particle swarm optimization scheme. *LNCS 4304* (2004), 868–873. 10, 11, 23

[50] Porter, M. An algorithm for suffix stripping. *Program 14*, 3 (1980), 130–137. 58

[51] R. Agrawal, J. Gehrke, D. G., and Raghavan, P. Automatic subspace clustering of high dimensional data. *Data Mining and Knowledge Discovery 11*, 1 (2005), 5–33. 2

[52] R. Mendes, J. K., and Neves, J. The fully informed particle swarm: Simpler, maybe better. *IEEE Trans. Evol. Comput. 8*, 3 (2004), 204C210. 10, 22, 23

[53] R. Tibshirani, G. Walther, e. a. Estimating the number of clusters in a dataset via the gap statistic. Tech. rep., Stanford Univeristy, 2000. 64

[54] S. Goil, H. N., and Choudhary, A. Mafia: Efficient and scalable subspace clustering for very large data sets. Tech. rep., Northwestern University, 1999. 2

[55] Salton, G., and Buckley, C. Term-weighting approaches in automatic text retrieval. *Information Processing and Management 24*, 5 (1988), 513–523. 56

[56] Suganthan, P. N. Particle swarm optimizer with neighborhood operator. In *CEC* (1999), p. 1958C1962. 10

[57] T. Pang-ning, S. M., and Vipin, K. *Introduction to Data Mining.* Pearson Education Inc, 2006. 57

[58] T. PERAM, K. V., AND MOHAN, C. K. Fitness-distance-ratio based particle swarm optimization. In *SIS* (2003), p. 174C181. 10

[59] TSENG, L. Y., AND YANG, S. B. A genetic approach to the automatic clustering algorithm. *Pattern Recognition 34*, 2 (2001), 415–424. 67

[60] VAN DEN BERGH, F., AND ENGELBRECHT, A. P. A cooperative approach to particle swarm optimization. *IEEE Trans. Evol. Comput. 8*, 3 (2004), 225C239. 11

[61] VAN DEN BERGH, F., AND ENGELBRECHT, A. P. Cooperative learning in neural networks using particle swarm optimizers. *South African Computer Journal 26* (2004), 84–90. 11, 39

[62] VAN DER PUTTEN, P., AND VAN SOMEREN (EDS), M. Coil challenge 2000: The insurance company case. Tech. rep., Published by Sentient Machine Research, 2000. 1, 54

[63] X. XIE, W. Z., AND YANG, Z. A dissipative particle swarm optimization. In *CEC* (2002), p. 1456C1461. 11

[64] X. ZHOU, X. HU, X. Z. X. L., AND SONG, I.-Y. Context-sensitive semantic smoothing for the language modeling approach to genomic ir. In *SIGIR* (2006). 60

[65] Y. SHI, R. E. A modified particle swarm optimizer. In *CEC* (1998), pp. 69–73. 9, 22

[66] Y. SHI, R. E. Particle swarm optimization with fuzzy adaptive inertia weight. In *Proc. Workshop Particle Swarm Optimization* (2001), p. 101C106. 9

[67] ZAIT, M., AND MESSATFA, H. A comparative study of clustering methods. *Future Generation Computer Systems 13*, 2-3 (1997), 149–159. 45

[68] ZHAO, Y., AND KARYPIS, G. Criterion functions for document clustering: Experiments and analysis. Tech. rep., Univ.of Minnesota, 2001. 59