USING LAGRANGIAN RELAXATION AND COLUMN GENERATION FOR DATA CLUSTERING

by

Mehmet Ayrancı

BS, Industrial Engineering, Bilkent University, 2006

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in FBE Program for Industrial Engineering

Boğaziçi University

2009

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Professor İ. Kuban Altınel for his excellent guidance, precious counseling, and inspiring supervision during the preparation of this dissertation. I appreciate his invaluable support and limitless encouragement.

I would like to thank Professor Ethem Alpaydın, and Necati Aras for taking time to examine this dissertation and for taking part in my thesis jury. I have very much benefited from their useful discussions.

I have very much benefited from the discussions with my dear friends in the Department of Industrial Engineering, among whom I would like to mention especially Burak Boyacı, M. Emre Keskin and Ünal Gök.

I also want to thank TÜBİTAK for its financial support.

This thesis is partly supported by Boğaziçi Research Fund Grant No:08A304D.

ABSTRACT

USING LAGRANGIAN RELAXATION AND COLUMN GENERATION FOR DATA CLUSTERING

Clustering is the organization of patterns, which are usually represented as vectors in multidimensional spaces, into a given number of groups with similar characteristics. It can be formulated as a mathematical optimization model whose objective is to locate cluster representatives so that the sum of dissimilarities with the representative and given data vectors is minimized. In this work, we first formulate clustering problem for the minimization of the total expected distances between cluster representatives and data vectors, assuming that dissimilarities are measured using a metric separable with respect to the coordinates. We then propose a Lagrangian relaxation scheme for the solution of the resulting mixed integer linear programming problem. Our second formulation is for any distance metric and is analogous to the formulation of the multi-facility Weber problem in many dimensions. The resulting nonconvex optimization problem is solved using column generation and d.c. programming techniques. According to our computational experiments we say that although the first one has low accuracy, the second one overperforms k-means algorithm.

ÖZET

LAGRANGE GEVŞETME VE SÜTUN ÜRETME KULLANARAK VERİ ÖBEKLEME

Veri öbekleme temel olarak benzer özellikleri olan ögeleri, kesişimleri boş olan verilen sayıda altkümelerde toplamayı amaçlar. Böylece her altküme tek bir temsilciyle ilişkilendirilebilir ve temsilcilerin belirlenmesi gündeme gelir. Her öge, her bileşeni sayısal değerler taşıyan bir özellik vektörü olarak gösterilir ve ögeler arası benzemezlik metrikler yardımıyla ölçülür. Veri öbekleme, amacı altkümeyi oluşturan özellik vektörleri ile temsilcileri arasındaki toplam benzemezliği enküçüklemek olan bir matematiksel eniyileme modeli olarak gösterilebilir.

Bu çalşmada, ilk olarak veri öbekleme problemini, benzemezliğin koordinatlara göre ayrılabilen bir metrik ile ölçüldüğünü varsayarak, özellik vektörleriyle temsilcileri arasındaki toplam beklenen uzaklığı enküçüklemek amacıyla gösterimliyoruz. Sonra ortaya çıkan karma tamsayılı doğrusal programlama problemini çözmek için bir Lagrange gevşetme yöntemi öneriyoruz. İkinci gösterimimiz ise herhangi bir benzemezlik metriği için kullanılabilmekte ve birçok yönden çok tesisli Weber problemine benzemektedir. Sonuçta oluşan dışbükey olmayan eniyileme problemini, düşey üretme ve d.c. programlama teknikleri ile çözüyoruz. Sayısal deneylerimize göre, ilk yöntemin düşük başarıya sahip olmasına rağmen ikinci yöntemin k-ortalama algoritmasından çok daha başarılı olduğunu söyleyebiliriz.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iii
ABSTRACT iv
ÖZET v
LIST OF FIGURES viii
LIST OF TABLES ix
LIST OF SYMBOLS/ABBREVIATIONS x
1. INTRODUCTION 1
2. PROBLEM FORMULATION 3
2.1. Basic Concepts and Notations 3
2.2. Mathematical Programming Formulations 4
3. LITERATURE REVIEW 8
4. CLUSTERING WITH LAGRANGIAN RELAXATION AND SUBGRADIENT OPTIMIZATION 12
4.1. Lagrangian Relaxation 12
4.2. Reductions for the Squared Euclidean Distance 15
4.3. Subgradient Optimization 17
5. CLUSTERING WITH DC PROGRAMMING AND COLUMN GENERATION 20
5.1. Equivalent Formulation 20
5.2. Column Generation and Pricing 21
5.3. Pricing Heuristics 25
5.4. New Clustering Algorithm 25
5.5. An Illustrative Example 28
6. COMPUTATIONAL EXPERIMENTS 32
6.1. Results for the Lagrangian Heuristic 32
6.2. Results for the Column Generation Algorithm 34
7. CONCLUSIONS 43
APPENDIX A: DERIVATIONS 47
A.1. Reductions for the Squared Euclidean Distance 47
A.2. Pricing as a DC Programming Problem 48

APPENDIX B: DISTORTION ERRORS 50
REFERENCES 55

LIST OF FIGURES

Figure 3.1. A taxonomy of clustering approaches 9

Figure 3.2. Algorithm of k-means clustering 9

Figure 4.1. Lagrangian heuristics 19

Figure 5.1. Pricing heuristics 26

Figure 5.2. Column generation algorithm 27

Figure 5.3. An illustrative example 29

Figure 6.1. An example of validation set: $K = 5$, size=100 and degree=$Easy$. 36

Figure 6.2. An example of training set: $K = 5$, size=100 and degree=$Easy$ . . 36

Figure 6.3. An example of validation set: $K = 9$, size=150 and degree=$Medium$ 37

Figure 6.4. An example of training set: $K = 9$, size=150 and degree=$Medium$ 37

Figure 6.5. An example of validation set: $K = 2$, size=50 and degree=$Difficult$ 38

Figure 6.6. An example of training set: $K = 2$, size=50 and degree=$Difficult$ 38

LIST OF TABLES

Table 6.1. Average of all combinations 33

Table 6.2. Percentage of deviation over cluster factor 33

Table 6.3. Percentage of deviation over size factor 33

Table 6.4. Percentage of deviation over dimension factor 34

Table 6.5. The results of data sets with Size=50 39

Table 6.6. The results of data sets with Size=100 40

Table 6.7. The results of data sets with Size=150 41

Table 6.8. The results for some UCI data sets 42

Table B.1. Distortion errors for m=2 50

Table B.2. Distortion errors for m=6 51

Table B.3. Distortion errors for m=10 52

Table B.4. Distortion errors for m=20 53

Table B.5. Distortion errors for m=30 54

LIST OF SYMBOLS/ABBREVIATIONS

$\boldsymbol{a}_j$	q dimensional feature vector j
a_{jk}	kth dimension of feature vector $\boldsymbol{a}_j$
c_t	Reduced cost of column t
m	Number of clusters
n	Number of feature vectors
q	Number of dimensions
$\boldsymbol{x}_i$	Codebook vector or center vector of cluster i
x_{ik}	kth dimension of cluster i
y_{ij}	Vector j is represented by codebook vector i
λ	Lagrangian multipliers
DC	Differences of convex functions
LB	Lower Bound
$RLMP$	Restricted linear master problem
UB	Upper bound
VQ	Vector quantization

1. INTRODUCTION

Clustering is the organization of patterns, which are usually represented as vectors in multidimensional spaces, into a given number of groups with similar characteristics. Vector quantization (VQ), a technique for clustering, is a representation of vectors that come from a probability density function or that is a part of a sample by a finite number of prototype vectors. In other words, VQ is dividing data into clusters, each of which represented by a prototype vector. Data can be transmitted over channels by transmitting only index of the prototype vector or the cluster for each data vector. Hence, a vector quantizer facilitates data compression which is the transfer of high rate discrete or analog data by a stream of reasonably low rate data for communication over a digital link.

The primary goal of the compression is to obtain minimum rate required for a given fidelity. The goal is satisfied when data is represented with the most convenient prototype vector in quantization. In other words, the fundamental aim of compression is to minimize the expected distance between prototype vector and data vectors.

In this work, VQ for clustering is evaluated from two different sides. First, it is assumed that data comes from a probability density function. Then, the minimization of expected distortion in VQ is handled using a mixed integer programming model. Since it is hard to solve such problems optimally in a reasonable time and cost, Lagrangian relaxation with subgradient optimization approach will be applied. Second, it assumed that data is taken from a finite sample instead of a density function. Then, the minimization of expected distortion transforms to a nonlinear integer optimization model which will be solved using column generation and differences of convex functions (DC) optimization.

This thesis is organized as follows. A new mathematical programming formulation is presented in Chapter 2 for clustering problems after introducing basic concepts and notations. In Chapter 3, clustering literature is visited to give a general view of cluster-

ing and to express important studies that facilitate this work. Chapter 4 introduces an approach to solve clustering by Lagrangian relaxation and subgradient optimization. From a different point of view, clustering problem is reformulated in Chapter 5, then a new algorithm that is composed of column generation and differences of convex functions (DC) programming problem is proposed to solve in an efficient way. Chapter 6 discusses applications of two different methods that is proposed Chapter 4 and Chapter 5, while comparing them with the k-means algorithm. Finally, Chapter 7 presents some concluding remarks and possible research directions.

2. PROBLEM FORMULATION

In this chapter, we introduce basic concepts and notations on vector quantization, and give problem formulation of the model.

2.1. Basic Concepts and Notations

In general, the aim of VQ is to determine a mapping from a continuous space to a discrete space. The continuous space can be obtained from continuous density function $f(\boldsymbol{a})$ of q-dimensional random vectors. On the other hand, a training sequence of n independent q-dimensional vectors can approximate the continuous space. The discrete space is a collection of a finite number of prototype or codebooks $\boldsymbol{x}_i = 1, 2, ..., m$. The codebook vectors represent the centroid of clusters, in cluster analysis literature.

VQ mainly consists of two mappings: an encoder and a decoder. Encoder assigns a channel index i for each input vector $\boldsymbol{a}_i \in \mathbb{R}^q$ that is an instance of a continuous space. Index is transferred over a digital channel. Then, decoder reproduces a vector $\boldsymbol{x}_i \in \mathbb{R}^q$ from an alphabet. Alphabet indicates which codebook will represent the input vector according to the index value.

In VQ, once codebooks or in other words mappings in the decoder are known, the representation of an input vector $\boldsymbol{a}$ is obtained by finding the codebook $\boldsymbol{x}_c$ closest to vector $\boldsymbol{a}$. However, approximation generates a cost because of the distortion of codebook $\boldsymbol{x}_c$ from the actual input vector $\boldsymbol{a}$. The distortion is represented by $d(\boldsymbol{x}_c, \boldsymbol{a})$. Many distortion measures have been proposed in the literature. The most common for reasons of mathematical convenience is the squared error distortion [1]:

$$d(\boldsymbol{x}_c, \boldsymbol{a}) = \sum_{k=1}^{q} (x_{ck}, a_k)^2 \tag{2.1}$$

However, the squared error criterion may give undesirable clusters which are the vector

groups represented by the codebooks. The squared error distortion tends to work well with isolated and compact clusters [2].

The index c (the index of the codebook) is defined as

$$c = \text{argmin}_{i=1,2,...,m}\{d(\boldsymbol{x}_i, \boldsymbol{a})\}$$

or

$$d(\boldsymbol{x}_c, \boldsymbol{a}) = \min_{i=1,2,...,m}\{d(\boldsymbol{x}_i, \boldsymbol{a})\}$$

where $\boldsymbol{x}_i = 1, 2, ..., m$ are the codebooks. Then VQ becomes optimal placement of the input data to codebooks, namely a placement which minimizes the expected reproduction or quantization error:

$$\Xi(x) = \int_{\boldsymbol{a}} d(\boldsymbol{x}_c, \boldsymbol{a}) f(\boldsymbol{a}) d\boldsymbol{a} \tag{2.2}$$

In fact, VQ is a competitive learning method that uses error minimization since we are trying the minimization of expected value in (2.2).

2.2. Mathematical Programming Formulations

Let $\{\boldsymbol{a}_j\}_{j=1}^n$ be a random sample of n independent q-dimensional vectors and express the expected quantization error as

$$\Xi(x) = \sum_{j=1}^{n} \int_{\boldsymbol{a}_j} d(\boldsymbol{x}_c, \boldsymbol{a}_j) f_j(\boldsymbol{a}_j) d\boldsymbol{a}_j, \tag{2.3}$$

after assuming that the coordinates of random vector $\boldsymbol{a}_j$ are distributed according to the joint probability distribution $f_j(\boldsymbol{a}_j)$. Here $d\boldsymbol{a}_j$ is a differential volume element in the q-dimensional hyperspace of vectors in which the integral is taken. The expected distortion is a function of the coordinates of the codebooks and can be interpreted as the

best expected distance between the codebooks and random vectors. Then, VQ is defined as the optimal placement of the codebooks, namely a placement which minimizes the expected distortion that becomes equivalent to the solution of the optimization problem

$$\Xi(x) = \min_{\boldsymbol{x}_1,...,\boldsymbol{x}_m} \sum_{j=1}^{n} \int_{\boldsymbol{a}_j} \min_{i=1,...,m} \{d(\boldsymbol{x}_i, \boldsymbol{a}_j)\} f_j(\boldsymbol{a}_j) d\boldsymbol{a}_j. \tag{2.4}$$

Consider the linear programming problem

$$\begin{aligned} \min \quad & \sum_{i=1}^{m} y_{ij} d(\boldsymbol{x}_i, \boldsymbol{a}_j) && (2.5)\\ s.t. \quad & \sum_{i=1}^{m} y_{ij} = 1 && (2.6)\\ & y_{ij} = 0, 1 \quad i = 1, ..., m, && (2.7) \end{aligned}$$

where $d(\boldsymbol{x}_i, \boldsymbol{a}_j)$ and y_{ij} are, respectively, the objective function coefficients and binary variables. The constraints in the above optimization problem describe a simplex whose vertices are the unit vectors and the zero vectors. As a result, the objective function is minimized if $y_{c_j j} = 1$ for $c_j = \arg\min_{i=1,...,m}\{d(\boldsymbol{x}_i, \boldsymbol{a}_j)\}$, while $y_{ij} = 0$ for $j \neq c_j$. Hence, this linear programming problem is equivalent to the term $\min_{i=1,...,m}\{d(\boldsymbol{x}_i, \boldsymbol{a}_j)\}$ of (2.4), and the minimization problem can be approximated by the following binary integer programming problem:

$$\begin{aligned} \min \quad & \sum_{j=1}^{n} \int_{\boldsymbol{a}_j} \sum_{i=1}^{m} y_{ij} d(\boldsymbol{x}_i, \boldsymbol{a}_j) f_j(\boldsymbol{a}_j) d\boldsymbol{a}_j && (2.8)\\ s.t. \quad & \sum_{i=1}^{m} y_{ij} = 1 \quad j = 1, ..., n && (2.9)\\ & y_{ij} = 0, 1 \qquad i = 1, ..., m; j = 1, ..., n. && (2.10) \end{aligned}$$

From a different point of view, let $\{\boldsymbol{a}_j\}_{j=1}^{n}$ be a training sample set of n independent q-dimensional vectors distributed according to the joint probability density

function, and then the aim is to minimize the distortion measure

$$\Xi(x) = \min_{\boldsymbol{x}_1,...,\boldsymbol{x}_m} \sum_{j=1}^{n} \min_{i=1,...,m} \{d(\boldsymbol{x}_i, \boldsymbol{a}_j)\} \tag{2.11}$$

after assuming that the training sample is large enough to represent the population. In other words, it is a clustering problem that searches for the cluster centers, $\boldsymbol{x}_i$, so that total distortion is minimized.

The term $\min_{i=1,...,m}\{d(\boldsymbol{x}_i, \boldsymbol{a}_j)\}$ in (2.11) is equivalent to the following integer linear programming problem

$$\min \quad \sum_{i=1}^{m} y_{ij} d(\boldsymbol{x}_i, \boldsymbol{a}_j) \tag{2.12}$$

$$s.t. \quad \sum_{i=1}^{m} y_{ij} = 1 \tag{2.13}$$

$$y_{ij} = 0, 1 \quad i = 1, ..., m \tag{2.14}$$

Then, the minimization problem (2.11) becomes,

$$\min \quad \sum_{i=1}^{m} \sum_{j=1}^{n} y_{ij} d(\boldsymbol{x}_i, \boldsymbol{a}_j) \tag{2.15}$$

$$s.t. \quad \sum_{i=1}^{m} y_{ij} = 1 \quad j = 1, ..., n \tag{2.16}$$

$$y_{ij} = 0, 1 \qquad i = 1, ..., m; j = 1, ..., n. \tag{2.17}$$

where

$$y_{ij} = \begin{cases} 1 & \boldsymbol{a}_j \text{ is in cluster } \quad i \\ 0 & \text{otherwise.} \end{cases}$$

It is possible to see that (2.15)-(2.17) is the well known multi-facility Weber problem where all customer demands are equal to 1 unit and dimension of the vectors

is possibly more than 2 using the analogies customer and sample, location vector and feature vector, facility and codebook, distance measure and distortion measure.

The difference of the mathematical programming formulations (2.8)-(2.10) and (2.15)-(2.17) is mainly in their objective functions. The first problem is obviously harder to solve from the second one as a consequence of the integration. However, it gives more reliable estimations since it is directly based on population distribution.

3. LITERATURE REVIEW

Clustering is a difficult problem to solve, and differences in assumptions and contexts in different scientific disciplines have made the transfer of useful generic concepts and methodologies slow to occur [3]. Different approaches to clustering data can be described with the help of the hierarchy shown in Figure 3.1. At the top level, there is a distinction between hierarchical and partitional approaches Hierarchical methods produce a nested series of partitions, while partitional methods produce only one. This study applies a partitional approach to clustering such as k-means algorithm. Jain et al. [3] presents an overview of pattern clustering methods from a statistical perspective, with a goal of providing useful advice and references to fundamental concepts accessible to the broad community of clustering practitioners.

Another dichotomy in clustering techniques is based on the strategy used while assigning patterns to clusters. A *hard* clustering algorithm allocates each pattern to a single cluster. However, *soft* clustering method assigns degrees of membership in several clusters to each input pattern. it can be converted to a *hard* clustering by assigning each pattern to the cluster with the largest measure of membership [3]. This conversion is applied in this work which will be explained in the following chapters.

Since similarity is fundamental to the definition of a cluster, a measure of the similarity between two patterns drawn from the same feature space is essential to most clustering procedures. Because of the variety of feature types and scales, the distance measure must be chosen carefully. It is most common to calculate the dissimilarity (or distortion) between two patterns using a distance measure defined on the feature space [3]. The most common and the simplest distortion measure is squared Euclidian distance. Moreover, different distortion measures used in literature are reported in [1] and [4].

In the area of clustering literature, a rather large number of methods exist. They all have similar goals but differ considerably in the way they work. A general view of

Figure 3.1. A taxonomy of clustering approaches

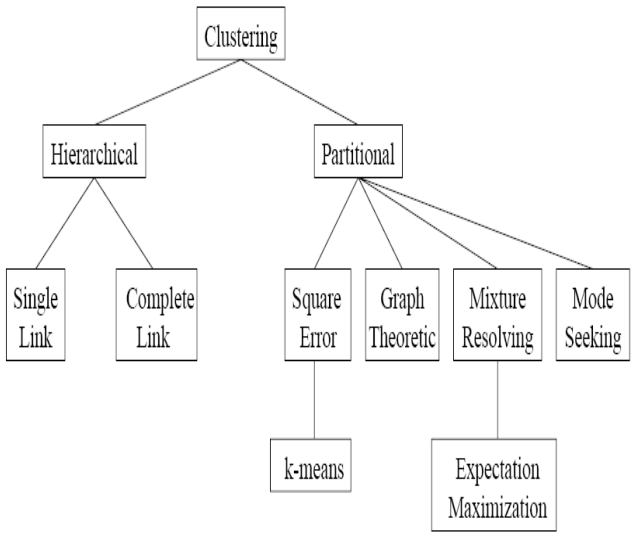


Figure 3.2. Algorithm of k-means clustering

1. Choose k cluster centers to coincide with k randomly-chosen patterns.
2. Assign each pattern to the closest cluster center.
3. Recompute the cluster centers using the current cluster memberships.
4. If a convergence criterion is not met, go to step 2. Typical convergence criteria are: no (or minimal) reassignment of patterns to new cluster centers, or minimal decrease in the total distortion.

these algorithms have been reported in [3], [5], [6], and [7]. The most popular algorithm in clustering is the k-means algorithm [8]. It is a method used to automatically partition a data set into k groups. It proceeds as algorithm in Figure 3.2. Several variants of the k-means algorithm have been reported in the literature. Jain et al. [3] gives a brief explanation of these variants.

Vector quantization (VQ) is a well-known clustering method. Vector quantization techniques are used mainly for data compression, which is a prerequisite in order to achieve better computer storage utilization and better bandwidth utilization in communications. VQ algorithms are intended to represent data by a reduced number of

elements that approximate the characteristics of the original data set as accurately as possible. Gray [2] gives a detailed review of vector quantization. Many VQ algorithms have been proposed such as the ones given in [1] and [9]. Mathematical programming models for VQ based clustering appears much more productive for applying optimization approaches. Mirkin [10] presents a comprehensive description of the current mathematical theories and many of the most recent developments in mathematical clustering.

In this work, clustering is formulated as the optimization problems explained in Chapter 2. Therefore, it is useful to report some techniques in the literature about solving optimization problems.

Many hard optimization problems can be viewed as a collection of simple subproblems combined by a relatively small set of complicating constraints. Dualizing the side constraints produces a Lagrangian subproblem whose optimal value is a lower bound (for minimization problems) on the optimal value of the original problem. This approach has led to improved algorithms for a number of difficult problems in the areas of routing, location, scheduling, assignment and set covering. A systematic development of Lagrangian relaxation as a means of exploiting special problem structure can be found in [11] and [12].

Given a set of points in the Euclidean plane, the multi-facility Weber problem consists of locating a given number of facilities, so that the sum of distances between each point and its nearest facility is minimized. It is known to be NP-hard even if the facilities are on a single line [13]. Starting from Cooper's seminal work [14] many exact and approximate solution methods have been proposed. Cooper's Alternate Location Allocation (ALA) is the earliest of them. It is originally supported for the case where the distance between the facilities and customers is measured using the Euclidean distance functions and consists of the alternating sequence of location and allocation phases starting at a randomly selected facility location. In the location phase, a Weber problem is solved using Weiszfeld's algorithm [15] for every facility with respect to their their own customer allocations. The allocation phase is simpler: customers are assigned

to the closest facility. It noticeable that k-means is a variant of ALA where customers have unit demands and squared Euclidean distance is used to measure distances instead of Euclidean one. The popularity of ALA is because of the simplicity and efficiency. However, its accuracy depends very much on the initial conditions. These are also two main characteristics of k-means.

Among the exact solution methods, Krau's algorithm [16] is the most efficient one. It is based on an equivalent set partitioning formulation and the use of column generation where the solution of nonconvex optimization problem for pricing is required and Foster and Ryan's branching rule [17] is used. Branch and price is used successfully for solving very large integer programming problems [18], and thus Krau's success is not surprising. Another more recent application of branch and price for the solution of the multi-facility Weber problem is due to Righini and Zaniboni [19]. They follow the line of research by Krau but they solve pricing subproblems using Chen's algorithm [20]. This is done by using an outer approximation algorithm [21] in Krau's method [16]. According to their computational experiments, using Chen's algorithm is especially more efficient when the number of facility versus the number of customer ratio is high.

Chen et al. [20] presents an application of DC programming problem to the multi-facility Weber problem. DC is a recent technique of global optimization that allows the solution of problems whose objective function and constraints can be expressed as a differences of two convex functions. A general view of DC programming problem is reported in [21], [22] and [23]. Chen et al. [20] reformulated Weber problem as DC programs which can be reduced to concave minimization problems over convex sets to apply DC to the multi-facility Weber problem.

4. CLUSTERING WITH LAGRANGIAN RELAXATION AND SUBGRADIENT OPTIMIZATION

The minimization problem (2.8)-(2.10) that is proposed in Chapter 2 is a nonlinear integer programming problem. Hence, it is hard to solve such problems optimally in a reasonable time for even small instances. Therefore, we propose a Lagrangian relaxation based subgradient optimization algorithm in the sequel.

4.1. Lagrangian Relaxation

Lagrangian relaxation is a method that relaxes some group of constraints and attaches multipliers to these constraints in order to bring them into the objective function. Its optimal value is a lower bound on the optimal value of the original minimization problem. Lagrangian relaxation can be also used as a base for accurate heuristics [12].

First, let us consider the case where the objective function is separable with respect to coordinates, namely

$$d(\boldsymbol{x}, \boldsymbol{a}) = \sum_{k=1}^{q} h_k(x_k, a_k) \tag{4.1}$$

with $h_k(x_k, a_k)$ is a function of the kth feature (i.e. the kth entry of the q-dimensional vectors $\boldsymbol{x}$ and $\boldsymbol{a}$). A particular example is the squared Euclidean distance, where

$$h_k(x_k, a_k) = (x_k - a_k)^2.$$

Then, the objective function (2.8) can be equivalently written as

$$\min_{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_m} \sum_{j=1}^{n} \int_{\boldsymbol{a}_j} \sum_{i=1}^{m} y_{ij} (\sum_{k=1}^{q} h_k(x_{ik}, a_{jk})) f_j(\boldsymbol{a}_j) d\boldsymbol{a}_j. \tag{4.2}$$

Let us assume that there are q copies of each codebook vector, one for each dimension of the future space, and only related entry is active. For example for the rth dimension

$$\sum_{k=1}^{q} h_k(x_{ik}, a_{jk}) = h_r(x_{ir}, a_{jr}). \tag{4.3}$$

Then, we can formulate the optimization problem (2.8)-(2.10) as

$$z_{IP} = \min \quad \sum_{i=1}^{m}\sum_{j=1}^{n} y_{ij}(\sum_{k=1}^{q} u_{ijk} \int h_k(x_{ik}, a_{jk}) f_{jk}(a_{jk}) da_{jk}) \tag{4.4}$$

$$s.t. \quad \sum_{i=1}^{m} y_{ij} = 1 \qquad j = 1, ..., n \tag{4.5}$$

$$\sum_{k=1}^{q} u_{ijk} = q y_{ij} \qquad i = 1, ..., m; j = 1, ..., n \tag{4.6}$$

$$y_{ij} = 0, 1 \quad u_{ijk} = 0, 1 \quad i = 1, ..., m; j = 1, ..., n; k = 1, ..., q. \tag{4.7}$$

Here

$$y_{ij} = \begin{cases} 1 & \text{if sample vector } j \text{ is presented by codebook } i, \\ 0 & \text{otherwise} \end{cases}$$

and

$$u_{ijk} = \begin{cases} 1 & \text{if coordinate } k \text{ of sample vector } j \text{ is presented by codebook } i, \\ 0 & \text{otherwise.} \end{cases}$$

The constraint set of above optimization problem (4.6) unifies the q copies of a codebook vector. Therefore, our assumption of copying codebooks becomes applicable on the optimization problem (2.8)-(2.10) to derive a new optimization problem. The main problem (2.8)-(2.10) or (4.4)-(4.7) is a large integer programming problem. Applying Lagrangian relaxation can enable decomposition into relatively small programming problems, so that the problem becomes solvable in a reasonable time.

The Lagrangian subproblem obtained after relaxing constraint set (4.5) is

$$\min \quad \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{q} y_{ij}(u_{ijk}\int h_k(x_{ik},a_{jk})f_{jk}(a_{jk})da_{jk}) + \sum_{i=1}^{m}\sum_{j=1}^{n}\lambda_j y_{ij} - \sum_{j=1}^{n}\lambda_j \tag{4.8}$$

$$s.t. \quad \sum_{k=1}^{q} u_{ijk} = qy_{ij} \qquad i=1,...,m; j=1,...,n \tag{4.9}$$

$$y_{ij}=0,1 \quad u_{ijk}=0,1 \quad i=1,...,m; j=1,...,n; k=1,...,q. \tag{4.10}$$

Here λ_j is the Lagrange multiplier. The last term in the objective function (4.8) is constant since the Lagrange multipliers are fixed. As a result, it becomes possible to decompose the subproblem with respect to codebooks and sample vectors as

$$z_{IP}^{ij}(\lambda) = \min \quad \sum_{k=1}^{q} y_{ij}(u_{ijk}\int h_k(x_{ik},a_{jk})f_{jk}(a_{jk})da_{jk}) + \lambda_j y_{ij} \tag{4.11}$$

$$s.t. \quad \sum_{k=1}^{q} u_{ijk} = qy_{ij} \tag{4.12}$$

$$u_{ijk}=0,1 \quad k=1,...,q \tag{4.13}$$

As it can be observed, when codebook i is not assigned to sample point j, namely when $y_{ij}=0$, objective function value is 0 for that codebook and sample vector. Otherwise, $y_{ij}=1$ and

$$\overline{z}_{IP}^{ij}(\lambda) = \min \sum_{k=1}^{q}\int h_k(x_{ik},a_{jk})f_{jk}(a_{jk})da_{jk} \tag{4.14}$$

has to be solved by eliminating the constraint set (4.12) of the optimization problem; because $y_{ij}=1$ implies that $u_{ijk}=1$ for all k. Therefore, this is simply equivalent to the minimization of the sum of the q 1-dimensional optimization problem

$$\overline{z}_{IP}^{ijk}(\lambda) = \min \int h_k(x_{ik},a_{jk})f_{jk}(a_{jk})da_{jk} \tag{4.15}$$

in x_{ik}. The optimal objective value of the Lagrangian subproblem becomes

$$z_{IP}(\lambda) = \sum_{i=1}^{m}\sum_{j=1}^{n} \min\left\{0, \sum_{k=1}^{q} \overline{z}_{IP}^{ijk}(\lambda) + \lambda_j\right\} - \sum_{j=1}^{n} \lambda_j. \tag{4.16}$$

In other words, the solution of the new formulation that is introduced in Chapter 2 reduces to the solution of q one dimensional optimization problem, which can be incorporated with a subgradient algorithm to obtain a lower bound and a heuristic solution providing an upper bound.

4.2. Reductions for the Squared Euclidean Distance

The integral part of the objective function has not been mentioned up to this point. For some of the instances it will not be possible to derive a closed form analytical expression for the integral. Therefore, the expectation value cannot be calculated. Fortunately, it is more suitable to calculate the expectation by making efficient assumptions on the distance function.

It is mentioned in the previous section that the Lagrangian relaxation is applicable where distortion measure $d(\boldsymbol{x}, \boldsymbol{a})$ is separable over dimensions as in the Equation (4.1). For this study, squared Euclidian distance measure is taken as a distortion measure since it is separable over dimensions. Therefore, $h_k(x_k, a_k)$ can be replaced by $(x_k - a_k)^2$ in the equations of previous section and the objective function (4.15) becomes

$$Q_{ijk} = \int (x_{ik} - a_{jk})^2 f_{jk}(a_{jk}) da_{jk}. \tag{4.17}$$

Assume a_{jk} is distributed with mean μ_{jk} and variance σ_{jk} as in the formulation of the problem in Chapter 2. After derivations that is shown in Appendix A.1, optimal objective value for (4.15) is

$$Q_{ijk}^{*} = \sigma_{_{jk}}^{2}, \tag{4.18}$$

which results in the following expression for the Lagrangian subproblem (4.16):

$$z_{IP}(\lambda) = \sum_{i=1}^{m}\sum_{j=1}^{n} \min\left\{0, \sum_{k=1}^{q} \sigma_{jk}^2 + \lambda_j\right\} - \sum_{j=1}^{n} \lambda_j \tag{4.19}$$

On the other hand, to remove the integral part, the formulation (2.15)-(2.17) can be used. Similar to (2.8)-(2.10), we can assume that there are q copies of each codebook vector, one for each dimension of the future space, and only related entry is active. Then, we can write the following equivalent of the optimization problem:

$$z_{IP} = \min \quad \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{q} u_{ijk} h_k(x_k, a_k) \tag{4.20}$$

$$s.t. \quad \sum_{i=1}^{m} y_{ij} = 1 \qquad j = 1, ..., n \tag{4.21}$$

$$\sum_{k=1}^{q} u_{ijk} = q y_{ij} \qquad i = 1, ..., m; j = 1, ..., n \tag{4.22}$$

$$y_{ij} = 0, 1 \; u_{ijk} = 0, 1 \quad i = 1, ..., m; j = 1, ..., n; k = 1, ..., q. \tag{4.23}$$

Here

$$h_k(x_k, a_k) = (x_k - a_k)^2,$$

$$y_{ij} = \begin{cases} 1 & \text{if sample vector } j \text{ is presented by codebook } i \\ 0 & \text{otherwise,} \end{cases}$$

and

$$u_{ijk} = \begin{cases} 1 & \text{if sample coordinates } k \text{ of a sample vector } j \text{ is presented by codebook } i \\ 0 & \text{otherwise.} \end{cases}$$

At the optimization problem (4.20)-(4.23), constraint set (4.21) can be the constraints for Lagrangian relaxation. After substituting the constraints into the objective function

with Lagrangian multipliers, the problem (4.20)-(4.23) becomes

$$z_{IP}(\lambda) = \min \quad \sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{q} u_{ijk}(x_{ik} - a_{jk})^2 + \sum_{i=1}^{m}\sum_{j=1}^{n} \lambda_j y_{ij} - \sum_{j=1}^{n} \lambda_j \tag{4.24}$$

$$s.t. \quad \sum_{k=1}^{q} u_{ijk} = q y_{ij} \qquad i = 1, ..., m; j = 1, ..., n \tag{4.25}$$

$$y_{ij} = 0, 1 \; u_{ijk} = 0, 1 \quad i = 1, ..., m; j = 1, ..., n; k = 1, ..., q. \tag{4.26}$$

After following the identical steps that is applied for the relaxed optimization problem (4.8)-(4.10), The optimal objective value of the Lagrangian subproblem is then

$$z_{IP}(\lambda) = \sum_{i=1}^{m}\sum_{j=1}^{n} \min \left\{ 0, \sum_{k=1}^{q} \overline{z}_{IP}^{ijk}(\lambda) + \lambda_j \right\} - \sum_{j=1}^{n} \lambda_j, \tag{4.27}$$

where

$$\overline{z}_{IP}^{ijk}(\lambda) = \min \quad (x_{ik} - a_{jk})^2. \tag{4.28}$$

Solution of the 1-dimensional optimization problem (4.28) is trivial, when derivative is taken with respect to x_{ik}. The optimal value equals to 0, since derivation $2(x_{ik} - a_{jk}) = 0$ leads to $x_{ik} = a_{jk}$ at the optimal value. It is expected to be zero; because there is no variance as in Equation (4.15).

4.3. Subgradient Optimization

Lagrangian relaxation of formulations quickly brings into mind an issue that is needed to be resolved. The issue is how to find an appropriate value for λ. The approach of the subgradient optimization is used for deciding Lagrange multipliers. Subgradient optimization is an iterative procedure which, from an initial set of multipliers, generates further multipliers in a systematic way.

The algorithm can be viewed as a procedure that maximizes the lower bound

obtained from the Lagrangian subproblem; and at the same time a procedure that tries to minimize the upper bound iteratively according to the relaxed solution. On the other hand, optimal solution may be attained if the lower and the upper bounds become equal.

First of all, the relaxed constraints in the optimization problems (4.4)-(4.7) and (4.20)-(4.23) are

$$\sum_{i=1}^{m} y_{ij} = 1 \quad j = 1, ..., n. \tag{4.29}$$

Then, the basic procedure can be stated as the algorithm given in Figure 4.1

Finding an initial upper bound and updating the upper bound using the relaxed solution necessitate some heuristics within the subgradient algorithm. The initial upper bound can be set arbitrarily large. The heuristics for updating the upper bound by utilizing the relaxed solution converts an infeasible allocation of instances to a feasible allocation. The heuristics first looks at feasible allocations and locates the codebooks. Then it determines the unique codebook of infeasible instances according to the closest distance. Finally, it updates the codebooks according to all allocations.

Figure 4.1. Lagrangian heuristics

1. Let π be a user defined parameter satisfying $0 < \pi \leq 2$. Initialize Z_{UB} from a heuristic for the optimization problem. Set an initial value to Lagrange multipliers λ_j such as 1 to each one.
2. Solve the relaxed problem with current multipliers (λ_j's) to get a solution (y_{ij}) of value Z_{LB}.
3. Find a solution to the original problem by a heuristic that utilizes the current relaxed solution. Update Z_{UB} if the new upper bound is less than the previous one (improvement of the upper bound).
   (a) If $Z_{UB} = Z_{LB}$, then stop the algorithm (Optimal solution is found).
   (b) Otherwise, continue with the next step.
4. Define the jth entry G_j of the subgradient $\boldsymbol{G}$ for the relaxed constraints evaluated at the current solution as

$$G_j = 1 - \sum_{i=1}^{m} y_{ij} \quad j = 1, ..., n \tag{4.30}$$

5. Define the step size T as

$$T = \pi(Z_{UB} - Z_{LB}) / \sum_{j=1}^{n} (G_j)^2. \tag{4.31}$$

6. Update λ_j as

$$\lambda_j = \max(0, \lambda_j + TG_j) \quad j = 1, ..., n. \tag{4.32}$$

7. If the number of steps exceeds a certain number, then STOP.
   Otherwise, go to Step 2 to resolve the relaxed problem with the new set of multipliers.

5. CLUSTERING WITH DC PROGRAMMING AND COLUMN GENERATION

In this part of the study, an equivalent formulation is introduced. New formulation is a very large integer programming problem. Then, a new algorithm is applied to solve the optimization programming problem efficiently. The algorithm mainly utilizes from two methods: column generation and differences of convex functions (DC) programming.

5.1. Equivalent Formulation

In Chapter 2, for solving clustering problems the following mathematical programming formulation is proposed:

$$\min \quad \sum_{i=1}^{m}\sum_{j=1}^{n} y_{ij} d(\boldsymbol{x}_i, \boldsymbol{a}_j) \tag{5.1}$$

$$s.t. \quad \sum_{i=1}^{m} y_{ij} = 1 \quad j = 1, ..., n \tag{5.2}$$

$$y_{ij} = 0, 1 \qquad i = 1, ..., m; j = 1, ..., n. \tag{5.3}$$

Here $\{\boldsymbol{a}_j\}_{j=1}^{n}$ is a training sample set of n independent q-dimensional vectors and

$$y_{ij} = \begin{cases} 1 & \boldsymbol{a}_j \text{ is in Cluster } \quad i \\ 0 & \text{otherwise.} \end{cases}$$

However, (5.1)-(5.3) is not suitable to apply column generation that is proposed in the next section. Since many combinatorial problems involving the partition of a set of elements into subsets have been successfully addressed by column generation [19]. The main idea is to reformulate the multi-facility Weber problem (MWP) as a variant of the set covering problem and to put all non-linearities into pricing subproblem. A set

covering reformulation of the MWP yields the following master problem.

$$z_{IP} = \min \quad \sum_{t=1}^{n_e} c_t y_t \tag{5.4}$$

$$s.t. \quad \sum_{t=1}^{n_e} b_{jt} y_t \geq 1 \quad j = 1, ..., n \tag{5.5}$$

$$\sum_{t=1}^{n_e} y_t = m \tag{5.6}$$

$$y_t = 0, 1 \qquad t = 1, ..., n_e. \tag{5.7}$$

Here

$$b_{jt} = \begin{cases} 1 & j \in S_t \\ 0 & \text{otherwise} \end{cases}$$

and

$$y_t = \begin{cases} 1 & \text{If} \quad S_t \quad \text{is} \quad \text{chosen} \\ 0 & \text{otherwise.} \end{cases}$$

It is indicated the number of total sets of all possible clusters by n_e. Then, $n_e = 2^n - 1$. S_t is the subset t of the instances. c_t is the cost of cluster t. The cost is calculated when the distortion measure is minimized over the vectors that belongs to cluster t. Finally, two above mathematical formulations (5.1)-(5.3) and (5.4)-(5.7) are equivalent.

5.2. Column Generation and Pricing

First of all, the linear relaxation of the master problem is obtained by replacing the set of constraints (5.7) with

$$1 \geq y_t \geq 0 \quad t = 1, ..., n_e. \tag{5.8}$$

The inequalities $1 \geq y_t$ are redundant, because $c_t \geq 0 \quad t = 1, ..., n_e$, the right hand sides of the cover inequality (5.5) are 1, and the coefficients b_{jt} are zero or one. The

relaxation provides a lower bound for the optimization problem for non integer optimal variable values. If the relevant optimal solution is integral, then the optimal solution is attained also for the original optimization problem. The solution for the optimization problem need much more time and effort. However, still the problem (5.4)-(5.7) can be solved optimally using a branch-and-price algorithms such as the one given in [19] and [16].

Since the number of columns in the optimization problem (5.4)-(5.7) grows exponentially with the number of instances n, column generation is applied to solve the linear programming relaxation. This is the well known Dantzig-Wolfe decomposition. One chooses to solve a large number of smaller size, typically well-structured, subproblems instead of solving the original problem whose size and complexity are beyond what can be solved within a reasonable amount of time [24].

A restricted linear master problem ($RLMP$) is solved on a limited subset of columns; using optimal dual values obtained in this way, additional columns with negative reduced cost are generated and inserted into the restricted linear master problem. The process is iterated until no column with negative reduced cost exists, in which the case optimum of the linear master problem has been found. The restricted linear problem is

$$z_{RLMP} = \min \quad \sum_{t=1}^{T} c_t y_t \tag{5.9}$$

$$s.t. \quad \sum_{t=1}^{n_e} b_{jt} y_t \geq 1 \quad j = 1, ..., n \quad : u_j \tag{5.10}$$

$$\sum_{t=1}^{n_e} y_t = m \quad : v \tag{5.11}$$

$$y_t \geq 0 \qquad t \in T. \tag{5.12}$$

Here T is number of current columns in the current problem that is less than or equal to n_e, and u_j and v are the dual variables of the above constraints.

The pricing problem consists in search for new columns with negative reduced

cost at each iteration of the column generation. The reduced cost of a generic cluster t is

$$\bar{c}_t = c_t - \sum_{j=1}^{n} b_{jt} u_j - v, \tag{5.13}$$

where

$$c_t = \min_x \sum_{j=1}^{n} b_{jt} d(\boldsymbol{x}, \boldsymbol{a}_j), \tag{5.14}$$

since c_t is the minimum cost of cluster t or column t. Column and cluster are identical terms for this optimization problem formulation. If a column is added to $RLMP$, in fact it means that a cluster alternative is added. To find the minimum reduced cost among all other columns that are not added to $RLMP$, the following minimization problem has to be solved:

$$\bar{c}_{t^*} = \min_t (c_t - \sum_{j=1}^{n} b_{jt} u_j - v). \tag{5.15}$$

This pricing problem is equal to

$$\bar{c}_{t^*} = \min_x \left\{ \sum_{j=1}^{n} f_j(x) - \sum_{j=1}^{n} g_j(x) \right\} - v \tag{5.16}$$

with

$$f_j(x) = d(\boldsymbol{x}, \boldsymbol{a}_j) - u_j \tag{5.17}$$

and

$$g_j(x) = \max \left[\begin{array}{cc} d(\boldsymbol{x}, \boldsymbol{a}_j) - u_j & 0 \end{array} \right]. \tag{5.18}$$

This derivation can be found in Appendix A.2. Since $f_j(x)$ and $g_j(x)$ are convex functions and also their summations are convex functions, the expression of the re-

duced cost given with Equality (5.16) is a differences of two convex functions (DC) programming problem.

On the other hand, (5.16) can be reformulated equivalently as the following concave minimization problem:

$$\min \quad r - \sum_{j=1}^{n} \max(d(\boldsymbol{x}, \boldsymbol{a}_j) - u_j, 0) - \sum_{j=1}^{n} u_j - v \tag{5.19}$$

$$s.t. \quad \sum_{j=1}^{n} d(x, a_j) - r \leq 0 \tag{5.20}$$

$$r \geq 0 \quad x \in H \tag{5.21}$$

using the approach provided by Chen et al. [20]. Here, H is the convex hull of the feature vectors, and r is an auxiliary variable.

The optimal solution of the DC programming problem (5.16) or concave minimization problem (5.19)-(5.21) decides whether the optimal solution of the master optimization problem is attained or a new column must be added. If $c_{t^*} \geq 0$, it means an optimal solution is attained. Otherwise, new columns must be added. Let x^* be an optimal solution of (5.16) or (5.19)-(5.21). Then the new column is:

$$b_{jt^*} = \begin{cases} 1 & \text{if} \quad d(\boldsymbol{x}, \boldsymbol{a}_j) < u_j \\ 0 & \text{otherwise} \end{cases} \tag{5.22}$$

for $j = 1, ..., n$ and

$$b_{(n+1)t^*} = 1 \tag{5.23}$$

for $j = n + 1$.

If $c_{t^*} < 0$, the new column $\boldsymbol{b}_{t^*}$ is attached to $RLMP$ and the process continues with the new $RLMP$ to compute new values for u_j and v.

5.3. Pricing Heuristics

Fortunately, it is not necessary to find an optimal solution of the pricing problem that selects best column for attaching. We need to add a column until nonnegative reduced cost is attained, but not the column with minimum reduced cost. The pricing problem is a global optimization problem, it is expensive and time consuming to attain optimal solution at each iteration. Furthermore, it is practical to add a column that has a negative reduced cost instead of adding a column with the smallest reduced cost.

Therefore, Krau's pricing heuristics [16] is applied not to solve a global optimization problem at each iteration and to attain a column with a negative reduced cost. The heuristic is applied while a negative reduced cost column is attainable with current optimal pricing primal and dual solutions. If heuristics fail to give a negative reduced cost column, then the global optimization problem is again used to add a new column.

Steps of the pricing heuristics are listed in Figure 5.1. This procedure starts with a column that has negative reduced cost and obtain a new column with smaller (more negative) reduced cost. This is not necessarily the smallest reduced cost obtainable at the current iteration.

5.4. New Clustering Algorithm

It is essential to organize new clustering algorithm using column generation and *DC* programming. The concepts that are proposed in the previous two sections can be combined into algorithm in Figure 5.2.

In the scope of this work, it is assumed that the number of clusters is given. Therefore, a method to estimate is not included in the algorithm.

Initialization part (Step 1) is achieved by using k-means as mentioned in Chapter 3. Clusters k-means gives are transformed into columns of the restricted master problem. Each distortion measure of the cluster becomes the cost multiplier in the

Figure 5.1. Pricing heuristics

1. Set $I_S = \emptyset$

   For all $j = 1, ..., n$

   do if $d(\boldsymbol{x}^*, \boldsymbol{a}_j) < u_j$, then set $\mathrm{I}_s \leftarrow \mathrm{I}_s \cup \{j\}$

   where $\boldsymbol{x}^*$ is the solution of the pricing optimization problem and I_S is a set for variables.

2. Solve the Weber problem

$$\min \sum_{j \in I_S} d(\boldsymbol{x}, \boldsymbol{a}_j)$$

   and, let x_S^* be an optimal solution for the above optimization problem.

3. Set $N = 0$

   For all $j = 1, ..., n$

   do begin

   if $j \in I_S$ then

   if $d(\boldsymbol{x}_S^*, \boldsymbol{a}_j) \geq u_j$ then set $\mathrm{I}_s \leftarrow \mathrm{I}_s - \{j\}$ and $N \leftarrow N + 1$.

   else

   if $d(\boldsymbol{x}_S^*, \boldsymbol{a}_j) < u_j$ then set $\mathrm{I}_s \leftarrow \mathrm{I}_s \cup \{j\}$ and $N \leftarrow N + 1$,

   where N is an integer.

4. If $N > 0$, then go to step 2

   Else, the column

$$(b_{1t^*}; b_{2t^*}; ...; b_{nt^*}; 1)$$

   with

$$b_{jt^*} = \begin{cases} 1 \quad if \quad d(\boldsymbol{x}, \boldsymbol{a}_j) < u_j \\ 0 \qquad \text{otherwise} \end{cases} \qquad j = 1, ..., n$$

   enters to restricted linear master problem.

Figure 5.2. Column generation algorithm

1. Determine an initial feasible restricted master problem.
2. Solve the current restricted master problem.
3. Apply the pricing heuristic using the current optimal solution of pricing optimization problem and dual variables of the current restricted master problem.
4. If pricing heuristics fail to attain a column with a negative reduced cost, then solve the pricing optimization problem that is a DC programming problem.
5. If optimal value of the pricing problem (minimum reduced cost) is negative, then

   Add a new column

   Go to Step 2

   Else

   Column generation terminates,

   Optimal value is reached for relaxed master problem,

   Go to next step.
6. If the solution of the relaxed master problem is non-integer, then

   Convert soft clustering to hard clustering.

   Apply k-means with taking the hard clustering solution as initial centers

   Otherwise, optimal solution of clustering problem is found.

objective function of the $RLMP$.

Since the pricing heuristics uses an optimal solution of the pricing optimization problem, Step 3 is skipped at the first run. Pricing optimization problem is solved directly without applying pricing heuristics.

In Step 2, $RLMP$ which is a linear optimization problem is solved. The important point is the non-convexity of the general clustering problem affects pricing subproblem. By heuristic solution to pricing optimization problem, time and cost consumed by the nonlinear optimization is reduced into a minimum level. In fact, if the $RLMP$ is solved without the relaxation of binary variables, the column generation algorithm gives the

optimal result for the clustering problem besides the pricing heuristics.

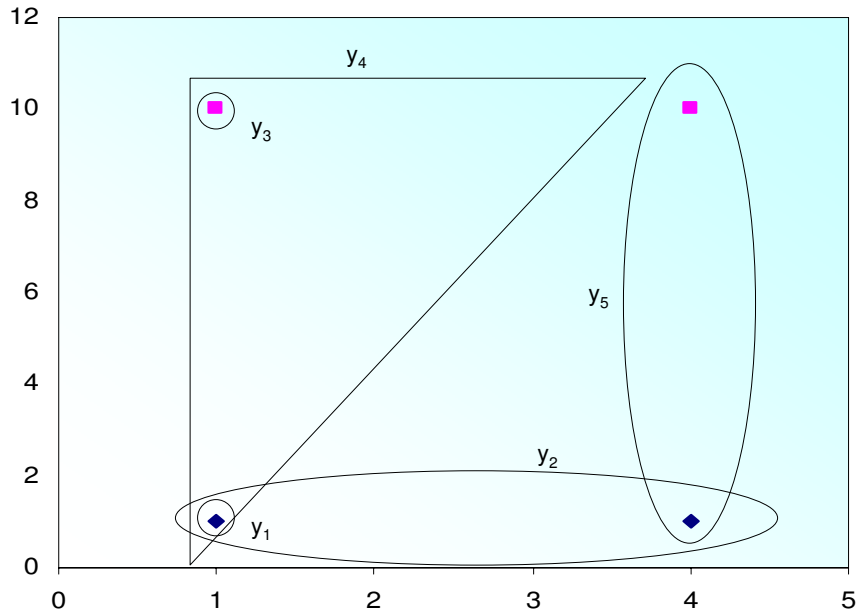
At the last step, if the solution of the relaxed master problem is non-integer, soft clustering result is attained. In the soft clustering solution of the algorithm, more than m clusters have a rate which is not 0 and it is smaller or equal to 1. It is not clear which m clusters are chosen. However, as a solution it is convenient to have m clusters which has a value of 1. Then, the solution is sorted in an increasing order. From the beginning of the solution, more deserving ones or in other words the larger ones is chosen ones as m clusters. This scheme converts soft clustering to hard clustering. However, if solution of the relaxed master problem is integer, hard clustering results is attained. There is no need for conversion from soft clustering to hard clustering. Furthermore, this means an optimal solution of general clustering problem is found.

5.5. An Illustrative Example

To motivate the general ideas of the new clustering algorithm, simple clustering example is presented. A plot of sample data is given in Figure 5.3; there are 4 points. It is desired to allocate 4 points into two clusters. Also, $m = 2$, $n = 4$ and $n_e = 15$ which is $2^4 - 1$. The data vector is:

$$a = \begin{bmatrix} 1 & 1 \\ 1 & 10 \\ 4 & 1 \\ 4 & 10 \end{bmatrix} \begin{matrix} \rightarrow a_1 \\ \rightarrow a_2 \\ \rightarrow a_3 \\ \rightarrow a_4 \end{matrix}.$$

Figure 5.3. An illustrative example



Furthermore, for initialization we assume that the following initial clusters and their costs are given:

$$y_1 : S_1 = \{a_1\} \rightarrow c_1 = 0$$
$$y_2 : S_2 = \{a_1, a_3\} \rightarrow c_2 = 4.5$$
$$y_3 : S_3 = \{a_2\} \rightarrow c_3 = 0$$
$$y_4 : S_4 = \{a_1, a_2, a_4\} \rightarrow c_4 = 60$$
$$y_5 : S_5 = \{a_3, a_4\} \rightarrow c_5 = 40.5.$$

Then, the restricted master linear optimization problem (5.9)-(5.12) becomes:

$$\begin{aligned}
\min \quad & 0y_1 + 4.5y_2 + 0y_3 + 60y_4 + 40.5y_5 \\
s.t. \quad & y_1 + y_2 + y_4 \geq 1 && : u_1 \\
& y_3 + y_4 \geq 1 && : u_2 \\
& y_2 + y_5 \geq 1 && : u_3 \\
& y_4 + y_5 \geq 1 && : u_4 \\
& y_1 + y_2 + y_3 + y_4 + y_5 = 2 && : v \\
& y_1, y_2, y_3, y_4, y_5 \geq 0.
\end{aligned}$$

The optimal objective value is 52.5, and corresponds to the optimal solution,

$$y_1 = 0, \quad y_2 = 0.5, \quad y_3 = 0.5, \quad y_4 = 0.5, \quad y_5 = 0.5$$

and

$$u_1 = 12, \quad u_2 = 12, \quad u_3 = 4.5, \quad u_4 = 48, \quad v = -12.$$

By solving DC programming problem, $c_{t^*} = -52.036$ and $x^* = (0, 8.782)$. Since the minimum reduced cost is negative, we need to add a column whose entries are determined as:

$$\begin{aligned}
d(x^*, a_1) = 61.563 > u_1 = 12 \quad & \rightarrow b_{1t} = 0 \\
d(x^*, a_2) = 2.482 < u_2 = 12 \quad & \rightarrow b_{2t} = 1 \\
d(x^*, a_3) = 76.569 > u_3 = 4.5 \quad & \rightarrow b_{3t} = 0 \\
d(x^*, a_4) = 17.482 < u_4 = 48 \quad & \rightarrow b_{4t} = 1
\end{aligned}$$

The new column is

$$b_6 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix},$$

which corresponds to the cluster which includes a_2 and a_4. The cost of the cluster is 4.5. After solving the $RMLP$ with applying column generation, the optimal solution

$$y_1 = 0, \quad y_2 = 1, \quad y_3 = 0, \quad y_4 = 0, \quad y_5 = 0, \quad y_6 = 1$$

is obtained. The corresponding dual optimal solution is

$$u_1 = 0, \quad u_2 = 0, \quad u_3 = 4.5, \quad u_4 = 4.5, \quad v = 0.$$

After finding $c_{t^*} = 0$, there is no need to add a new column, since c_{t^*} is nonnegative. The algorithm terminates with integer optimal solution, therefore the optimal solution

$$S_2 = \{a_1, a_3\} \quad \text{and} \quad S_6 = \{a_2, a_4\},$$

of clustering problem is found as the solution of its linear programming relaxation.

6. COMPUTATIONAL EXPERIMENTS

In order to measure the performance of proposed algorithms (clustering with Lagrangian relaxation and subgradient optimization, clustering with column generation and DC programming) and to compare with other known algorithms, data sets are created or taken from known test libraries and the algorithms are coded. The results obtained with the new algorithms are compared with the ones calculated using k-means.

6.1. Results for the Lagrangian Heuristic

In this section, clustering with Lagrangian relaxation and subgradient optimization that is proposed in Chapter 4 is considered. First of all, it is implemented by using MATLAB that provides a high-level programming language, an interactive technical computing environment, and functions for algorithm development, data analysis and visualization and numeric computation [25].

It is more convenient to create the data by defining multivariate density functions so that we can see the deviation from the actual one. The data creation is achieved by multivariate density function that randomly assigns a mean value for each cluster and dimension between 0 and 1000 and assigns a random standard deviation between 0 and 10. For each instance, a cluster is assigned randomly and it is produced by the mean and the standard deviation of that cluster. Data is created according to three properties: the number of instances (n), the number of dimensions (q) and the number of clusters (m). For the comparison, the combinations of the following values are used:

$$\begin{aligned} \text{Cluster} &= [\ \ 2 \quad 6 \quad 10 \quad 20 \quad 30 \ \ ]; \\ \text{Dimension} &= [\ \ 2 \quad 10 \quad 25 \quad 50 \quad 100 \ \ ]; \\ \text{Size} &= [\ \ 20 \quad 50 \quad 100 \quad 1000 \ \ ]. \end{aligned}$$

Table 6.1. Average of all combinations

min error	aver. error	max error	LB	UB
5.71E+07	1.82E+08	3.23E+08	0.00	7.77E+08

Table 6.2. Percentage of deviation over cluster factor

Cluster	aver. error	max error	UB
2	0.0000	0.0000	0.9992
6	0.9865	0.9939	0.9965
10	0.7455	0.8726	0.9427
20	0.6252	0.7631	0.8936
30	0.4999	0.6676	0.8575

In the comparison, squared Euclidean distance measure is determined as the distortion error measure. For each combination, k-means algorithm is applied with 100 replications. Minimum, average and maximum values that are achieved by k-means are stated for each combination. Finally, Lagrangian relaxation and subgradient optimization algorithm is applied for each combination. As a result, a lower bound (LB) and upper bound (UB) is attained.

The average distortion measure of all combinations are in Table 6.1. Then, distortion deviations form the minimum distortion error that is achieved by k-means in 100 replications are calculated by (error-minimum error)/error. The results of deviations according to cluster factor are in Table 6.2, according to input size factor are in Table 6.3 and according to dimension factor are in Table 6.4. Distortion error of each combination is presented in Appendix B.

Table 6.3. Percentage of deviation over size factor

Size	aver. error	max error	UB
20	0.8949	0.9535	0.9804
50	0.7309	0.8705	0.9369
100	0.7960	0.8951	0.9520
1000	0.6944	0.8277	0.9296

Table 6.4. Percentage of deviation over dimension factor

Dimension	aver. error	max error	UB
2	0.9180	0.9774	0.9976
10	0.7446	0.8917	0.9713
25	0.7870	0.8858	0.9656
50	0.7431	0.8634	0.9382
100	0.6701	0.8083	0.9145

First, we look at the overall picture to evaluate the algorithms. Unfortunately, the Lagrangian lower bounds do not mean anything; because most of the time, it is zero or close to zero. We can have this conclusion without the Lagrangian heuristics by identifying that distortion error must be greater than zero. The reason for low LB is the decomposition of the problem into subproblems. The minimization of subproblems are easy and do not need to satisfy a lot of constraints where Lagrangian multipliers cannot punish objective function effectively. On the other side the upper bounds are also much more bigger than k-means errors. Since the relaxation cannot find high lower bounds, upper bounds that corporate with the lower bounds, is not good enough. On the other hand, distortion error of k-means has a large variance depending on the initialization.

6.2. Results for the Column Generation Algorithm

In this section, clustering with column generation and DC that is proposed in Chapter 5 is considered with Algorithm 5.2. First of all, the algorithm is implemented using MATLAB and an .exe file that is written by [26] to achieve the differences of convex functions (DC) programming. To solve linear optimization problems with column generation, MOSEK optimization toolbox [27] is integrated into MATLAB which makes possible to call highly efficient MOSEK optimization engine from the MATLAB environment.

Experimental data sets categorized into two classes: synthetic data set that is created in this study and data set obtained from UCI repository [28]. To create syn-

thetic data, three different properties of data set are considered: the size of the training data set, number of clusters and distribution of data points according to easiness for clustering. Three different size of training data sets are taken: 50, 100 and 150 number of instances. Three different number of clusters are chosen: 2, 5 and 9. As a last property, data sets are categorized as easy, medium and difficult according to distribution of instances for clustering. On the other hand, the size of the validation sets are constant for all data sets which is 1000.

Since there are 3 property that each have 3 different values. Here are 27 experiment combinations. For each combination, 10 different data sets are created. Then, for each data set, 30 replications are achieved each with a different randomly generated initialization. For example, for a combination where the size of the training set is 50 ($Size = 50$), number of cluster is 5 ($K = 5$), and data set is categorized as medium (Medium), 10 different training and validation data sets are created. On each data set, the proposed algorithm is applied 30 times to avoid the randomness of k-means initialization. Different combinations of data sets are visualized in Figure 6.1, Figure 6.2, Figure 6.3, Figure 6.4, Figure 6.5, and Figure 6.6.

In the experimental design, the dimension of the data set is assumed 2. The reason is very fast increase in the execution time of the algorithm with the increase of dimension. Since the pricing problem which determines the new column for generation is a DC programming problem, it takes much time for higher dimensions.

In this study, k-means is compared with the proposed algorithm. At first, k-means is applied on the training set and then the column generation algorithm is applied after taking the k-means solution as an initial solution. The result of the algorithm is considered as initial centers of k-means, then again k-means is applied. Then, two results are compared on the validation sets via distortion measure which is the squared Euclidean distance. For each run, a value for improvement is attained by (error of k-means - error of the proposed algorithm)/error of k-means. The average values of improvements for 30 replications of each combination with 10 different data sets are shown in Table 6.5, Table 6.6, and Table 6.7 with the standard deviation in

Figure 6.1. An example of validation set: $K = 5$, size=100 and degree=$Easy$

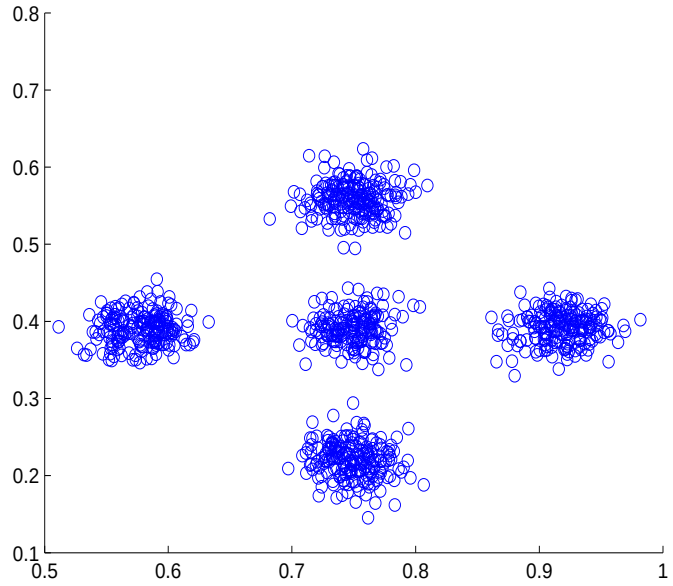


Figure 6.2. An example of training set: $K = 5$, size=100 and degree=$Easy$

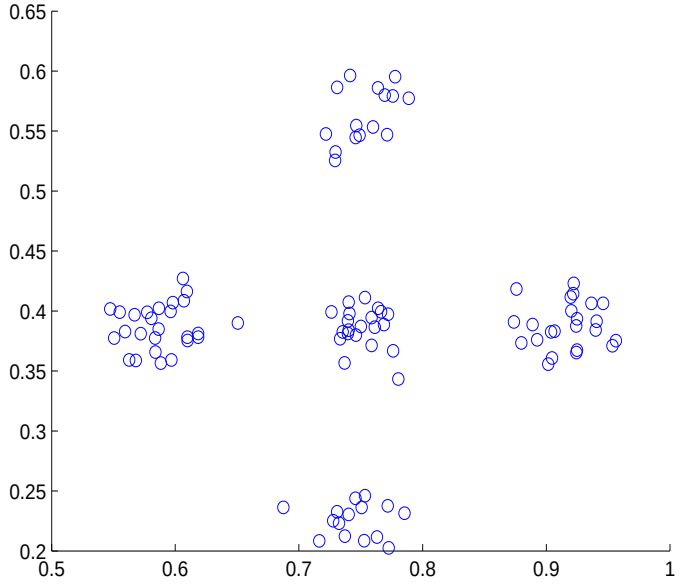


Figure 6.3. An example of validation set: $K = 9$, size=150 and degree=$Medium$

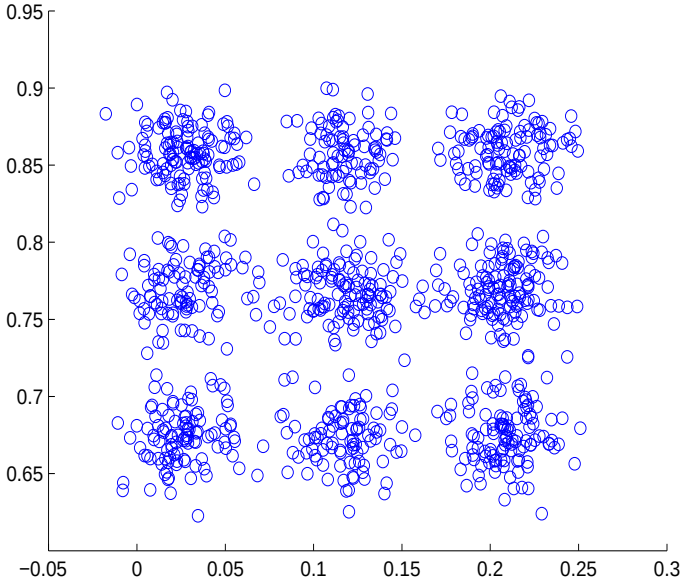


Figure 6.4. An example of training set: $K = 9$, size=150 and degree=$Medium$

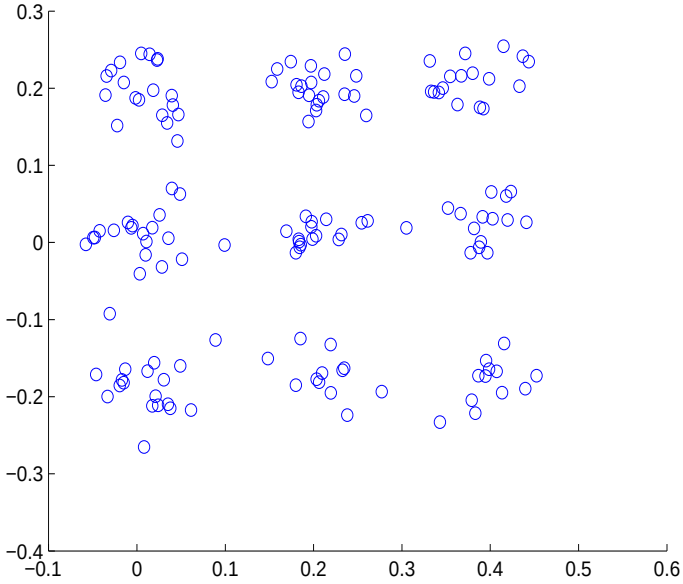


Figure 6.5. An example of validation set: $K = 2$, size=50 and degree=$Difficult$

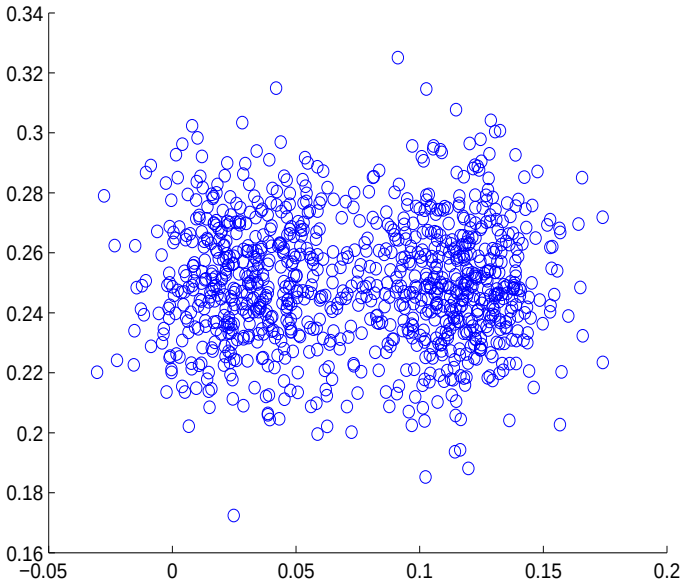


Figure 6.6. An example of training set: $K = 2$, size=50 and degree=$Difficult$

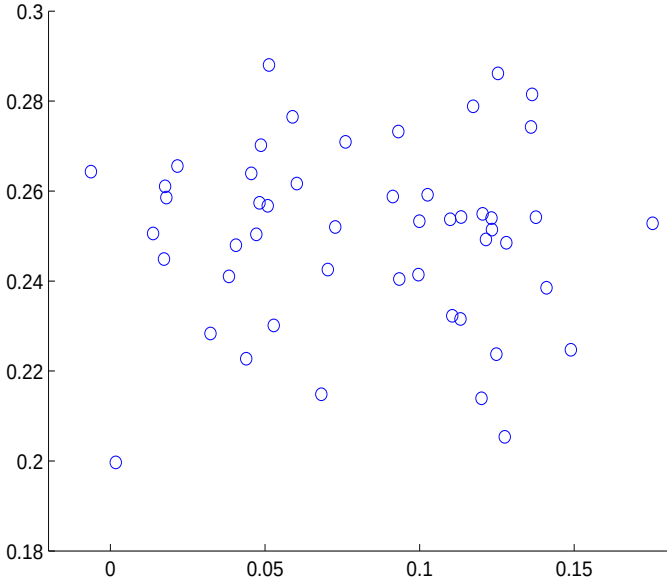


Table 6.5. The results of data sets with Size=50

Easy		**Medium**		**Difficult**	
K = 2					
0.000±0	0.000±0	0.000±0	0.000±0	0.000±0	-0.0119±0.014
0.000±0	0.000±0	0.000±0	0.000±0	0.000±0	0.000±0
0.000±0	0.000±0	0.000±0	0.000±0	0.000±0	0.000±0
0.000±0	0.000±0	0.000±0	0.000±0	0.000±0	0.000±0
0.000±0	0.000±0	0.000±0	0.000±0	-0.007±0.009	0.000±0
K = 5					
0.090± 0.166	0.045±0.140	0.041±0.073	**0.047±0.092**	**0.064±0.108**	0.029±0.059
0.063±0.144	0.039±0.148	0.000±0	0.036±0.072	**0.068±0.108**	**0.165±0.065**
0.076±0.156	**0.127±0.138**	**0.048±0.058**	**0.085±0.013**	**0.014±0.022**	**0.047±0.068**
0.109± 0.163	0.038±0.145	**0.027±0.046**	**0.056±0.078**	**0.087±0.110**	**0.025±0.043**
0.058± 0.115	0.083±0.160	**0.030±0.078**	0.032±0.112	**0.026±0.040**	0.010±0.076
K = 9					
0.188±0.119	0.019±0.110	0.027±0.083	0.032±0.083	**0.039±0.068**	**0.062±0.044**
0.203±0.110	0.022±0.078	**0.110±0.106**	**0.068±0.094**	**0.084±0.052**	**0.051±0.051**
0.089±0.108	0.046±0.096	**0.040±0.049**	**0.080±0.071**	**0.118±0.056**	**0.087±0.086**
0.065±0.094	**0.315±0.103**	**0.057±0.052**	**0.089±0.077**	-0.038±0.051	0.026±0.124
0.117±0.114	**0.224±0.107**	**0.030±0.039**	**0.087±0.071**	**0.056±0.060**	**0.053±0.062**

the replications. The bold results in these tables mean that it can be claimed that there is a significant improvement introduced by the new algorithm on k-means at the confidence level of 0.01. The paired hypothesis test is performed between errors of k-means and the proposed algorithm.

It is obvious that there is not an improvement when the cluster size is 2. K-means is very successful by itself. However, it is not the same for larger cluster sizes, since average improvement is nearly 10 percent when the number of clusters is 5 and 20 percent when the number of clusters is 9. The improvement value increases parallel to the increase in the number of cluster.

The proposed algorithm can reach to 50 percent improvements especially when data set is categorized as easy. K-means can find cluster centers that does not reflect the structure of the data set due to the initialization of k-means. This is punished

Table 6.6. The results of data sets with Size=100

Easy		**Medium**		**Difficult**	
K = 2					
0.000±0	0.000±0	0.000±0	0.000±0	0.000±0	0.000±0
0.000±0	0.000±0	0.000±0	0.000±0	0.000±0	0.000±0
0.000±0	0.000±0	0.000±0	0.000±0	0.000±0	0.000±0
0.000±0	0.000±0	0.000±0	0.000±0	0.000±0	0.000±0
0.000±0	0.000±0	0.000±0	0.000±0	0.000±0	0.000±0
K = 5					
0.253±0.368	**0.337±0.371**	0.000±0	0.000±0	**0.018±0.018**	**0.567±0.300**
0.211±0.357	**0.247±0.353**	0.000±0	0.062±0.190	**0.077±0.111**	0.039±0.096
0.343±0.346	**0.228±0.354**	**0.129±0.160**	0.000±0	0.012±0.063	**0.087±0.197**
0.222±0.298	**0.288±0.304**	**0.054±0.122**	0.029±0.089	0.067±0.127	0.012±0.066
0.262±0.353	**0.185±0.341**	0.044±0.113	0.000±0	0.04±0.093	0.017±0.052
K = 9					
0.59±0.323	**0.365±0.298**	**0.04±0.072**	0.000±0	-0.003±0.051	**0.135±0.125**
0.556±0.260	**0.502±0.284**	**0.091±0.064**	**0.224±0.169**	**0.072±0.048**	**0.196±0.180**
0.609±0.004	**0.632±0.175**	**0.063±0.058**	**0.28±0.191**	**0.091±0.097**	**0.181±0.120**
0.562±0.264	**0.±0.261**	-0.020±0.057	**0.21± 0.206**	**0.105±0.107**	**0.168±0.106**
0.484±0.254	**0.515±0.290**	**0.073±0.076**	**0.275±0.191**	**0.118±0.098**	**0.237±0.156**

much in the data sets which are categorized as easy, since features are divided obviously between each other. Furthermore, the increase in improvement value is observed with the increase in size of the data.

On the other hand, execution time of the algorithm increases with the increase in the size of the data set as expected. When the size of data arises to 200 or more, the execution time may reach up to days for some cases. However, it is not observed that the change in the number of clusters significantly effects the execution time of the algorithm.

Another experimental data set category in this study is the sets obtained from UCI repository; 7 data sets are chosen. The properties of data set is shown in Table 6.8. As it is explained in the beginning of this section, execution time for data sets more than two dimensions is not considerable. However, chosen data sets are more than

Table 6.7. The results of data sets with Size=150

Easy		**Medium**		**Difficult**	
K = 2					
0.000±0	0.000±0	0.000±0	0.000±0	0.000±0	0.000±0
0.000±0	0.000±0	0.000±0	0.000±0	0.000±0	0.000±0
0.000±0	0.000±0	0.000±0	0.000±0	0.000±0	0.000±0
0.000±0	0.000±0	0.000±0	0.000±0	0.000±0	0.000±0
0.000±0	0.000±0	0.000±0	0.000±0	0.000±0	0.000±0
K = 5					
0.292±0.391	**0.261±0.377**	0.078±0.274	**0.283±0.333**	0.000±0	0.042±0.159
0.205±0.347	**0.091±0.178**	**0.102±0.231**	0.000±0	**0.086±0.166**	0.000±0
0.274±0.372	**0.194±0.357**	**0.105±0.272**	**0.162±0.300**	0.022±0.085	**0.049±0.111**
0.188±0.348	**0.315±0.388**	0.073±0.224	0.115±0.236	**0.067±0.136**	0.021±0.113
0.177±0.322	0.161±0.333	**0.167±0.283**	0.097±0.254	0.000±0	**0.043±0.132**
K = 9					
0.502±0.258	**0.491±0.280**	**0.384±0.206**	**0.389±0.183**	**0.167±0.114**	**0.697±0.202**
0.486±0.303	**0.486±0.304**	**0.331±0.244**	**0.315±0.232**	**0.184±0.139**	**0.279±0.182**
0.404±0.316	**0.441±0.296**	**0.364±0.231**	0.131±0.234	**0.121±0.124**	0.025±0.101
0.111±0.252	**0.457±0.316**	0.036±0.106	**0.165±0.469**	**0.124±0.118**	**0.171±0.133**
0.489±0.307	**0.349±0.334**	**0.282±0.256**	**0.089±0.226**	**0.194±0.116**	**0.135±0.110**

two dimensions. Therefore, before applying the algorithm, dimensions of the data sets are reduced to two using the principal component analysis [6]. The experiments are performed on the reduced data sets including the error measurements. The proportion column in Table 6.8 explains in what proportion that reduced data reflects the properties of real data set.

On the other hand, K-fold cross validation is applied on the first three data sets (Lenses, Soybean, Zoo and Hayes-roth data sets) because of the number of instances. In the K-fold cross validation, the data set is divided randomly into K equal parts. To generate each pair, one of the K parts is kept out as the validation set, and the remaining $K-1$ pairs are combined to form the training set. For each fold of data set, 30 replications are applied as previous runs. Then, the average improvements of folds are reported in Table 6.8. The improvement is attained by (error of k-means algorithm - error of the proposed algorithm)/error of k-means algorithm as in the previous data

Table 6.8. The results for some UCI data sets

	Name	Instances	Dimension	Cluster	Proportion	Improvement
1	Lenses	24	4	3	0.7475	**0.1078±0.256**
2	Soybean	47	35	4	0.7064	0.0586±0.158
3	Zoo	101	17	7	0.8373	**0.2330±0.381**
4	Hayes-roth	132	5	3	0.9984	**0.0264±0.022**
5	Iris	150	4	3	0.9776	0.07969±0.200
6	Wine	178	13	3	0.9998	**0.0335±0.058**
7	Syn. Cont.	600	60	6	0.7714	**0.3257±0.205**

set category.

For the other data sets, data is divided into two sets: training and validation. Algorithm is applied on the training data sets with 30 replications. Distortion measure is applied to the validation data sets. The average results for improvement is reported in Table 6.8. Moreover, The bold results in Table 6.8 mean that it can be claimed that there is a significant difference between algorithms at the confidence level of 0.01. The paired hypothesis test is performed between errors of k-means algorithm and the proposed algorithm.

The improvement values are really high for higher number of clusters as concluded before. Improvement values of Zoo data set which has 7 clusters and Synthetic Control data set which has 6 clusters are 23 and 33 percent respectively. Furthermore, the proposed algorithm is more prosperous than k-means algorithm according to paired hypothesis test results. High improvement rates over k-means are expected since the proposed algorithm utilizes from optimization techniques. However, execution time that obligate the proposed algorithm is prohibitive for higher dimensions and major data sets is a disadvantage.

7. CONCLUSIONS

In this work, clustering is evaluated from two different perspectives using mathematical principles. First, it is assumed that data comes from a probability density function. Then, the minimization of expected distortion in VQ is handled using a nonlinear integer optimization. Since it is hard to solve such problems optimally in a reasonable time and cost, Lagrangian relaxation with the subgradient optimization approach is applied to solve nonlinear integer optimization problem. The algorithm gives upper bound and lower bound solutions as close as the optimal point.

Second, it assumed that data is taken from a finite sample instead of a density function. Then, the minimization of expected distortion transforms to a linear integer optimization by an equivalent formulation. New formulation is a very large integer programming problem. Then, a new algorithm is applied to solve the optimization programming problem efficiently after a linear relaxation of variables. The algorithm mainly utilizes column generation. However, to generate optimum columns, pricing is achieved due to differences of convex functions (DC) programming. Fortunately, it is not necessary to find the optimal solution of the pricing problem that selects best column for attaching. Krau's pricing heuristics is applied at each iteration and to attain a column with a negative reduced cost.

In order to measure the performance of two algorithms proposed in this study (clustering with Lagrangian relaxation and subgradient optimization, clustering with column generation and DC programming) and to compare with other known algorithms, data sets are created or taken from known data sets and implementation of algorithms on a soft environment are achieved.

Unfortunately, the Lagrangian lower bounds do not mean anything; because most of the time, it is zero or close to zero. The reason for low LB is the decomposition of the problem to subproblems. The minimization of subproblems are easy and do not need to satisfy a lot of constraints. Lagrangian multipliers cannot punish objective

function effectively. On the other side the upper bounds are also much more larger than k-means errors. Since the relaxation cannot find high lower bounds, upper bounds that corporate with the lower bounds, is not good enough. Lagrangian multipliers could not succeed the unification of subproblems. The gap between lower and upper bounds are high that is not prosperous to determine the optimal objective value. Therefore, it is not a practical strategy to apply Lagrangian relaxation on the clustering problem. However, if more strict subproblems are constructed that punish objective function effectively by Lagrangian multipliers, the optimal value of objective function can be interpreted in a more accurate way due to lower and upper bounds.

Furthermore, column generation algorithm is compared with k-means via distortion measure which is squared Euclidean. The paired hypothesis test is performed between errors of k-means and the proposed algorithm to claim that there is a significant difference between algorithms at the confidence level of 0.01.

It is obvious that there is not an improvement when the cluster size is 2. K-means is very successful by itself. It is an easy problem for k-means. Most of the time, the optimal value is found by a heuristic method, k-means. Therefore, the proposed algorithm could not ensure an improvement on k-means. However, it is not the same for larger cluster sizes. The improvement value increases parallel to the increase in the number of clusters. Such clustering problems are hard ones for k-means. It is an expected result, since the proposed algorithm utilizes techniques of optimization methods. Although the proposed algorithm is not an exact optimal solution procedure, it approaches to the optimal value. On the other hand, the increase in the number of clusters does not effect the proposed algorithm structure, only a right hand size of a constraint changes. Therefore, increase in the number of clusters is not a trouble for the proposed algorithm. Fortunately, the increase in the number of clusters ensures an advantage over other algorithms since the increase may effect them very much.

The proposed algorithm can reach to 50 percent improvements especially when data set is categorized as easy. K-means can find cluster centers that does not reflect the structure of the data set. If initial centers of k-means is located to exterior points,

it is not easy to get them to actual cluster locations. However, the proposed algorithm is not effected from the initials. It is an interesting remark that solutions of the proposed algorithm approach to a value for 30 replications most of the time during the experiments despite the unsteady solution of k-means for the initialization of the proposed algorithm.

Furthermore, the increase in improvement is observed with the increase in size of the data. However, execution time of the algorithm increase with the increase in the size of the data set as expected. At large data sets, k-means finds cluster centers in an easy way without expending much effort. Therefore, the results of the k-means is not close to optimal ones. However, the case is reverse for the proposed algorithm. Finally, according to our computational experiments it is significant that the column generation algorithm overperforms k-means as an overall picture.

The relaxation in master problem of this study provides a lower bound for the optimization problem. The solution for the original clustering optimization problem needs much more time and effort. However, still the relaxed problem can be solved optimal by using branch-and-price algorithms as a future work. This algorithm needs to solve column generation algorithm more than once until the optimal solution is reached. However, it does not seem applicable due to the execution time.

Execution time that makes the proposed algorithm non-applicable for higher dimensions and major data sets is a disadvantage. The time should be decreased. In higher dimensions, to avoid high execution times, improvements on DC programming problem can be applied. The global optimization problem takes much time for higher dimensions. This causes the proposed algorithm to be non-applicable for higher dimensions. Since the DC programming problem is tried to be solved optimally, this strategy can be changed. Heuristics can be applied on DC program to decrease the execution time.

Pricing heuristics continues until reduced cost is smaller than a value that is too close to 0. However, the convergence of reduced cost to for example 1 is fast, but the

decrease of the reduced cost from 1 to 0 is very low. The profit is not parallel to the execution time. There may be a heuristic for pricing that considers both the increase in execution time and the decrease in the reduced cost. Therefore, the algorithm stops earlier without much loss of information. This could clear off the disadvantage of the algorithm on the major data sets.

For major data sets, a heuristic could be constructed that utilizes from a minor part of data which is sufficient to signify properties of the original data set. Most of the time, properties of major data set could be indicated by a smaller part of the major data set by eliminating redundancies. First of all, minor data set is obtained and the proposed algorithm is applied on the minor part. Then the resulting cluster centers could be located for the major data set. These heuristics can eliminate the disadvantage of the execution time.

APPENDIX A: DERIVATIONS

A.1. Reductions for the Squared Euclidean Distance

Let

$$Q_{ijk} = \int (x_{ik} - a_{jk})^2 f_{jk}(a_{jk}) da_{jk} \tag{A.1}$$

and assume that a_{jk} is distributed by mean μ_{jk} and variance σ_{jk}. Then,

$$Q_{ijk} = \int (x_{ik}^2 - 2x_{ik}a_{jk} + a_{jk}^2) f_{jk}(a_{jk}) da_{jk} \tag{A.2}$$

which is equivalent to

$$Q_{ijk} = x_{ik}^2 \int f_{jk}(a_{jk}) da_{jk} - 2x_{ik} \int a_{jk} f_{jk}(a_{jk}) da_{jk} + \int a_{jk}^2 f_{jk}(a_{jk}) da_{jk} \tag{A.3}$$

and hence

$$Q_{ijk} = x_{ik}^2 - 2x_{ik}\mu_{jk} + \sigma_{jk}^2 + \mu_{jk}^2 \tag{A.4}$$

follows. When we apply optimality conditions and take the derivative with respect to x_{ik} to find an optimal solution in the minimization problem (4.15) we obtain

$$2x_{ik} - 2\mu_{jk} = 0. \tag{A.5}$$

Hence $x_{ik}^* = \mu_{jk}$ is an optimal solution with

$$Q_{ijk}^* = \mu_{jk}^2 - 2\mu_{jk}\mu_{jk} + \sigma_{jk}^2 + \mu_{jk}^2 \tag{A.6}$$

as the optimal value. Notice that this is equivalent to

$$Q^*_{ijk} = \sigma^2_{jk}. \tag{A.7}$$

A.2. Pricing as a DC Programming Problem

The reduced cost can be formulated as

$$\bar{c}_{t^*} = \min_t (c_t - \sum_{j=1}^{n} b_{jt} u_j - v) \tag{A.8}$$

where

$$c_t = \min_x \sum_{j=1}^{n} b_{jt} d(x, a_j). \tag{A.9}$$

Therefore,

$$\bar{c}_{t^*} = \min_t (\min_x \sum_{j=1}^{n} b_{jt} d(x, a_j) - \sum_{j=1}^{n} b_{jt} u_j - v), \tag{A.10}$$

which is equivalent to

$$\bar{c}_{t^*} = \min_{x, b_{jt}} \sum_{j=1}^{n} b_{jt} (d(x, a_j) - u_j) - v \tag{A.11}$$

from which

$$\bar{c}_{t^*} = \min_x \sum_{j=1}^{n} \min((d(x, a_j) - u_j, 0) - v \tag{A.12}$$

and thus

$$\bar{c}_{t^*} = \min_x \sum_{j=1}^{n} \{[(d(x, a_j) - u_j] - \max[d(x, a_j) - u_j, 0]\} - v \tag{A.13}$$

follows. the last expression can be written as,

$$\bar{c}_{t^*} = \min_{x} \left\{ \sum_{j=1}^{n} f_j(x) - \sum_{j=1}^{n} g_j(x) \right\} - v \tag{A.14}$$

with

$$f_j(x) = d(x, a_j) - u_j \tag{A.15}$$

and

$$g_j(x) = \max \left[\; d(x, a_j) - u_j, \quad 0 \; \right]. \tag{A.16}$$

Each of the function is convex and the pricing subproblem is a DC programming problem as a consequence.

APPENDIX B: DISTORTION ERRORS

Table B.1. Distortion errors for m=2

Dimension	Size	min	avg	max	LB	UB
q=2	n=20	1.09E+03	1.09E+03	1.09E+03	0.00	2.24E+06
q=2	n=50	2.31E+03	2.31E+03	2.31E+03	0.00	3.22E+05
q=2	n=100	4.34E+03	4.34E+03	4.34E+03	0.00	7.78E+06
q=2	n=1000	3.59E+04	3.59E+04	3.59E+04	0.00	1.96E+07
q=10	n=20	6.24E+03	6.24E+03	6.24E+03	0.00	6.42E+06
q=10	n=50	2.05E+04	2.05E+04	2.05E+04	0.00	1.41E+07
q=10	n=100	4.23E+04	4.23E+04	4.23E+04	0.00	4.97E+07
q=10	n=1000	3.21E+05	3.21E+05	3.21E+05	0.00	5.06E+08
q=25	n=20	1.69E+04	1.69E+04	1.69E+04	0.00	1.61E+07
q=25	n=50	4.18E+04	4.18E+04	4.18E+04	0.00	4.43E+07
q=25	n=100	8.31E+04	8.31E+04	8.31E+04	0.00	9.40E+07
q=25	n=1000	8.16E+05	8.16E+05	8.16E+05	0.00	1.31E+09
q=50	n=20	3.53E+04	3.53E+04	3.53E+04	0.00	3.56E+07
q=50	n=50	7.70E+04	7.70E+04	7.70E+04	0.00	1.10E+08
q=50	n=100	1.52E+05	1.52E+05	1.52E+05	0.00	1.93E+08
q=50	n=1000	1.62E+06	1.62E+06	1.62E+06	0.00	1.78E+09
q=100	n=20	5.54E+04	5.54E+04	5.54E+04	0.00	5.69E+07
q=100	n=50	1.64E+05	1.64E+05	1.64E+05	0.00	1.73E+08
q=100	n=100	3.53E+05	3.53E+05	3.53E+05	0.00	4.28E+08
q=100	n=1000	3.17E+06	3.17E+06	3.17E+06	0.00	4.11E+09
Average		3.50E+05	3.50E+05	3.50E+05	0.00	4.48E+08

Table B.2. Distortion errors for m=6

Dimension	Size	min	avg	max	LB	UB
q=2	n=20	1.80E+03	1.16E+05	1.26E+06	0.00	2.55E+06
q=2	n=50	2.36E+03	3.68E+05	2.68E+06	0.00	3.72E+06
q=2	n=100	4.74E+03	8.87E+05	4.40E+06	0.00	1.41E+07
q=2	n=1000	5.12E+04	5.13E+06	1.19E+07	0.00	8.17E+07
q=10	n=20	3.26E+03	1.47E+06	3.48E+06	0.00	6.58E+06
q=10	n=50	1.49E+04	7.42E+06	1.80E+07	0.00	2.90E+07
q=10	n=100	3.46E+04	1.47E+07	3.77E+07	0.00	7.52E+07
q=10	n=1000	2.94E+05	1.15E+08	3.49E+08	0.00	8.94E+08
q=25	n=20	8.09E+03	1.94E+06	6.81E+06	0.00	1.64E+07
q=25	n=50	3.14E+04	1.30E+07	3.59E+07	0.00	7.68E+07
q=25	n=100	7.18E+04	3.09E+07	6.63E+07	0.00	1.70E+08
q=25	n=1000	8.70E+05	3.92E+08	7.48E+08	0.00	1.89E+09
q=50	n=20	3.98E+06	1.50E+07	3.48E+07	0.00	4.52E+07
q=50	n=50	7.66E+04	3.48E+07	1.01E+08	0.00	1.40E+08
q=50	n=100	1.52E+05	6.86E+07	1.62E+08	0.00	2.58E+08
q=50	n=1000	1.71E+06	1.10E+09	2.20E+09	0.00	3.57E+09
q=100	n=20	5.23E+04	2.90E+07	6.36E+07	0.00	8.07E+07
q=100	n=50	4.32E+07	9.28E+07	2.04E+08	0.00	2.89E+08
q=100	n=100	3.07E+05	2.03E+08	4.05E+08	0.00	6.70E+08
q=100	n=1000	3.23E+06	1.89E+09	4.39E+09	0.00	7.30E+09
Average		2.71E+06	2.01E+08	4.42E+08	0.00	7.80E+08

Table B.3. Distortion errors for m=10

Dimension	Size	min	avg	max	LB	UB
q=2	n=20	4.44E+03	4.06E+04	1.64E+05	0.00	8.86E+05
q=2	n=50	1.34E+04	2.07E+05	6.52E+05	0.00	4.63E+06
q=2	n=100	1.67E+04	3.60E+05	2.43E+06	0.00	9.31E+06
q=2	n=1000	3.29E+05	8.80E+06	3.30E+07	0.00	2.15E+08
q=10	n=20	6.13E+05	2.21E+06	4.47E+06	0.00	6.64E+06
q=10	n=50	1.38E+04	4.94E+06	8.78E+06	0.00	2.36E+07
q=10	n=100	2.50E+06	1.21E+07	2.92E+07	0.00	5.82E+07
q=10	n=1000	2.25E+07	9.52E+07	2.51E+08	0.00	7.92E+08
q=25	n=20	9.30E+03	2.04E+06	5.97E+06	0.00	1.33E+07
q=25	n=50	4.50E+06	1.82E+07	3.03E+07	0.00	6.88E+07
q=25	n=100	1.10E+07	3.82E+07	7.12E+07	0.00	1.54E+08
q=25	n=1000	7.98E+05	3.45E+08	7.41E+08	0.00	1.78E+09
q=50	n=20	4.00E+06	1.13E+07	2.82E+07	0.00	3.45E+07
q=50	n=50	6.62E+04	3.57E+07	7.00E+07	0.00	1.35E+08
q=50	n=100	1.51E+05	7.79E+07	1.91E+08	0.00	3.35E+08
q=50	n=1000	2.41E+08	7.55E+08	1.91E+09	0.00	3.46E+09
q=100	n=20	3.53E+04	2.07E+07	4.08E+07	0.00	6.73E+07
q=100	n=50	2.15E+07	9.20E+07	1.81E+08	0.00	2.78E+08
q=100	n=100	5.41E+07	2.22E+08	3.95E+08	0.00	6.86E+08
q=100	n=1000	5.37E+08	1.79E+09	3.06E+09	0.00	7.58E+09
Average		4.50E+07	1.77E+08	3.53E+08	0.00	7.85E+08

Table B.4. Distortion errors for m=20

Dimension	Size	min	avg	max	LB	UB
q=2	n=20	0.00E+00	0.00E+00	0.00E+00	0.00	3.77E+05
q=2	n=50	2.85E+04	1.33E+05	3.07E+05	0.00	4.13E+06
q=2	n=100	9.12E+04	1.86E+05	3.08E+05	0.00	8.86E+06
q=2	n=1000	6.31E+05	2.65E+06	9.72E+06	0.00	1.53E+08
q=10	n=20	0.00E+00	0.00E+00	0.00E+00	0.00	1.93E+06
q=10	n=50	7.63E+05	2.53E+06	5.53E+06	0.00	1.92E+07
q=10	n=100	1.55E+06	7.54E+06	1.32E+07	0.00	5.96E+07
q=10	n=1000	2.46E+07	1.00E+08	2.05E+08	0.00	7.58E+08
q=25	n=20	0.00E+00	0.00E+00	0.00E+00	0.00	4.86E+06
q=25	n=50	6.00E+06	1.34E+07	2.39E+07	0.00	5.60E+07
q=25	n=100	1.89E+07	3.35E+07	5.11E+07	0.00	1.63E+08
q=25	n=1000	1.39E+08	3.54E+08	6.23E+08	0.00	1.92E+09
q=50	n=20	0.00E+00	0.00E+00	0.00E+00	0.00	1.78E+07
q=50	n=50	1.30E+07	3.01E+07	5.71E+07	0.00	1.14E+08
q=50	n=100	3.22E+07	8.19E+07	1.55E+08	0.00	2.94E+08
q=50	n=1000	4.55E+08	1.12E+09	1.76E+09	0.00	3.94E+09
q=100	n=20	0.00E+00	0.00E+00	0.00E+00	0.00	3.44E+07
q=100	n=50	3.95E+07	8.96E+07	1.52E+08	0.00	2.39E+08
q=100	n=100	5.53E+07	1.93E+08	3.86E+08	0.00	6.02E+08
q=100	n=1000	9.41E+08	2.58E+09	3.86E+09	0.00	7.86E+09
Average		8.64E+07	2.31E+08	3.65E+08	0.00	8.12E+08

Table B.5. Distortion errors for m=30

Dimension	Size	min	avg	max	LB	UB
q=2	n=50	1.51E+03	2.36E+04	6.07E+04	0.00	3.32E+06
q=2	n=100	3.84E+04	1.21E+05	2.69E+05	0.00	9.06E+06
q=2	n=1000	4.28E+05	1.50E+06	7.36E+06	0.00	1.55E+08
q=10	n=50	7.30E+05	1.76E+06	3.44E+06	0.00	1.55E+07
q=10	n=100	2.59E+06	6.05E+06	1.00E+07	0.00	5.11E+07
q=10	n=1000	6.24E+07	9.47E+07	1.60E+08	0.00	7.81E+08
q=25	n=50	1.64E+06	5.50E+06	1.21E+07	0.00	4.42E+07
q=25	n=100	7.98E+06	2.69E+07	4.98E+07	0.00	1.33E+08
q=25	n=1000	1.46E+08	3.12E+08	4.96E+08	0.00	1.88E+09
q=50	n=50	3.36E+06	1.72E+07	3.92E+07	0.00	7.99E+07
q=50	n=100	2.60E+07	7.38E+07	1.50E+08	0.00	2.70E+08
q=50	n=1000	3.74E+08	1.07E+09	1.60E+09	0.00	3.89E+09
q=100	n=50	6.91E+04	4.14E+07	9.59E+07	0.00	1.72E+08
q=100	n=100	5.69E+07	1.60E+08	2.58E+08	0.00	5.39E+08
q=100	n=1000	1.58E+09	2.72E+09	3.93E+09	0.00	7.88E+09
Average		1.51E+08	3.02E+08	4.54E+08	0.00	1.06E+09

REFERENCES

1. Linde, Y., A. Buzo, and R. M. Gray, "An Algorithm for Vector Quantizer Design", *IEEE Transactions on Communications*, Vol. 28, 1980.

2. Gray, R. M., "Vector Quantization", *IEEE ASSP Magazine*, pp. 4–29, April 1984.

3. Jain, A. K., M. N. Murty, and P. J. Flynn, "Data Clustering: A Review", *ACM Computing Surveys*, Vol. 31, pp. 264–323, 1999.

4. Makur, A., "Improved Distortion Maeasures for Image Vector Quantization", Department of Electrical Communication Engineering, Indian Institute of Science, Bangalore, India.

5. Jain, A. K. and R. C. Dubes, *Algorithms For Clustering Data*, Prentice Hall Advanced References Series, 1988.

6. Alpaydın, E., *Introduction to Machine Learning*, The MIT Press, 2004.

7. Fritzke, B., "Some Competitive Learning Methods", April 1997, systems Biophysics, Institute for Neural Computation, Ruhr-Universat Bochum.

8. MacQueen, J., "Some Methods for Classification and Analysis of Multiattribute Instances", *Proceedings of the Fifty Berkeley Symposium on Mathematics*, Vol. 1, pp. 281–296, 1967.

9. Equitz, W., "A New Vector Quantizition Clustering Algorithm", *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. 37, pp. 1568–1575, 1989.

10. Mirkin, B., *Mathematical Classification and Clustering*, Kluwer Academic, 1996.

11. Geoffrion, A. M., "Lagrangean Relaxations for Integer Programming", *Mathematical Programming Study*, pp. 82–114, 1974.

12. Fisher, M. L., "The Lagrangian Relaxation Method for Solving Integer Programming Problems", *Management Science*, Vol. 27, pp. 1–18, 1981.

13. Sherali, H. D. and F. L. Nordai, "NP-Hard, Capacitated, Balanced p-Median Problems on a Chain Graph with a Continuum of Link Demands", *Mathematics of Operations Research*, Vol. 13, pp. 32–49, 1988.

14. Cooper, L., "The Transportation-Location Problem", *Operations Research*, pp. 94–108, 1972.

15. Weiszfeld, E., "Sur le Point Lequel la Somme des Distances de n Points Donné Est Minimum", *Tohoku Math Journal*, Vol. 43, pp. 355–386, 1937.

16. Krau, S., "Extensions du probléme de Weber", 1997, ph.D. Thesis, Department of Mathematics and Industrial Engineering, École Polytechnique de Montréal.

17. Ryan, D. M. and B. A. Foster, "An Integer Programming Approach to Scheduling", *Computer Scheduling of Public Transport*, pp. 269–289, 1981.

18. Barnhart, C., E. L. Johnson, G. L. Nemhauser, M. Savelsberg, and P. H. Vance, "Branch-and-price: Column Generation for Solving Huge Integer Programs", *Operations Research*, Vol. 46, 3, pp. 316–329, 1998.

19. Righini, G. and L. Zaniboni, "A Branch-and-price Algorithm for the Multi-source Weber Problem", *Internal Journal of Operational Research*, Vol. 2, pp. 188–207, 2007.

20. Chen, P.-C., P. Hansen, B. Jaumard, and H. Tuy, "Solution of the Multisource Weber and Conditional Weber Problems by DC Programming", *Operations Research*, Vol. 46, 4, pp. 548–562, 1998.

21. Horst, R. and H. Tuy, *Global Optimization Deterministic Approaches*, Springer-Verlag.

22. An, L. T. H. and P. D. Tao, "The DC Programming and DCA Revisited with DC Models of Real World Nonconvex Optimization Problems", *Annals of Operations Research*, Vol. 133, pp. 23–46, 2005.

23. Horst, R., P. M. Pardolos, and N. V. Thoai, *Introduction to Global Optimization*, Kluwer Academic.

24. Vanderback, F. and M. Savelsberg, "A Generic View of Dantzig-Wolfe Decomposition in Mixed Integer Programming", *Operations Research Letters*, Vol. 34, pp. 296–306, 2006.

25. "MATLAB", `http://www.mathworks.com/`.

26. Boyacı, B., "Private Communication", Industrial Engineering Department, Boğaziçi University, İstanbul.

27. "The Mosek Optimization Toolbox", `http://www.mosek.com`.

28. Asuncion, A. and D. J. Newman, "UCI Machine Learning Repository", 2007, `http://www.ics.uci.edu/ mlearn/MLRepository.html`.