University of Alberta

A Parameter Selection Framework for Semi-Supervised Clustering Algorithms

by

Mojgan Pourrajabi

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

Abstract

Many clustering techniques require parameter settings and depending on an algorithms sensitivity to the parameter, the choice of the parameter value can be very important. Several approaches have been proposed to find the "best" value of the clustering parameter for the different unsupervised clustering methods.
We introduce a general method, denoted as "Cross-validation framework for finding clustering parameters" (CVCP). Given a data set, CVCP selects the "best" parameter value for a semi-supervised clustering method based on available constraints or labels that are given as input to a semi-supervised clustering method. CVCP is evaluated based on selecting the "best" value of k for a semi-supervised Kmeans-based clustering algorithm and the "best" value of MinPts for a semi-supervised density-based clustering algorithm. Our experimental results show that using the framework to select parameters can significantly improve the expected performance of a semi-supervised clustering method when appropriate parameter values often have to be "guessed".

Acknowledgements

I would like to express my gratitude to my supervisor, Professor Joerg. Sander for his invaluable guidance, encouragement and patience throughout this thesis. I cannot imagine this project done without his incredible support.

Furthermore, I would like to thank Professor Ricardo. Campello, Arthur Zimek and Davoud Moulavi at University of Alberta for their precious useful comments, remarks in this project.

Finally, I would like to thank my family for their endless love and being supportive in making my life easy in graduate school.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Cluster analysis is one of the significant tasks in data mining. Cluster analysis divides data into groups (clusters) that should be meaningful and useful. Often cluster analysis is employed as a pre-processing step for further understanding the data. Cluster analysis captures the natural structure of the data and prepares it for a variety of data analyses in other domains. Many fields benefit from the extraction of useful knowledge about the data: psychology and other social sciences, biology, statistics, information retrieval and pattern recognition.

Cluster analysis offers many applications to practical problems. Clustering methods have been used in biology to analyze the large amounts of genetic information [1]. Extracting climate patterns in the atmosphere and oceans helps understanding the Earth's climate better. Business owns huge data bases of information on current and potential customers or stocks. Clustering can be used to segment customers into a small number of groups for additional analysis and marketing activities [1].

Back to the definition of cluster analysis, we will define cluster analysis more concisely illustrating why cluster analysis can be challenging. Cluster analysis groups objects based on only information found in the data that describe objects and their relationships. One of the goals of clustering is to provide a partitioning of data so that objects residing within one group (cluster) are similar to one another and different from the objects of the other groups. The greater the similarity (or homogeneity) within a group and the greater the difference between groups, the better or more distinct the clustering.

It is important to mention the fact that the notion of a cluster is not well defined and the notion of clustering can vary for different applications. Figure 1.1 shows twenty points and three different ways of dividing them into clusters [1]. Different clusters have different shape of markers. Figures 1.1(b), 1.1(c) and 1.1(d) show that one data set can be partitioned

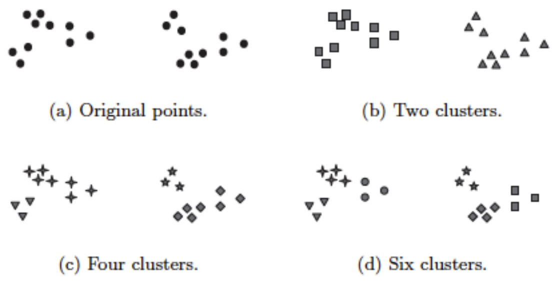


Figure 1.1: Different ways of clustering the same points [1].

in different ways.

1.1 Unsupervised Clustering

Clustering is usually performed when no information is available concerning the membership of data items to predefined classes. For this reason, clustering is traditionally seen as part of unsupervised learning. Here, we introduce three major unsupervised clustering paradigms that are available in the literature.

1.1.1 K-means

K-means [2] is one of the simplest unsupervised clustering algorithms. Prototype-based clustering techniques create a one-level partitioning of data objects. K-means defines a partition in terms of a centroid that is usually represented by the mean of a group of points and typically applied to points in a continuous n-dimensional space.
The formal description of the K-means clustering is as follows:

Definition 1. Given a set of objects $x_1, x_2, .., x_n$, where each object is from a continuous n-dimensional space, k-means clustering aims to partition objects into k groups $(k \leq n)$ $G = g_1, g_2, ..., g_k$ to minimize the square distances between objects and corresponding centroids:

$$\arg_g \min \sum_{i=1}^{k} \sum_{x_j \in g_i} \|x_j - \mu_i\|^2$$

where μ_i is the mean of the cluster g_i.

K-means Algorithm: The procedure follows a simple idea to cluster a given data set into a certain number of clusters (assume k clusters) fixed a priori. K-means needs k initial centroids, where k is a user specified parameter, to create k clusters. In the first iteration of the algorithm, each point is assigned to the closest initial cluster centroid and the group of points that are assigned to a centroid is a cluster. The centroid of a cluster will be updated by the mean of the points residing in that cluster. The assignment and update steps will be repeated until no point changes its cluster. The algorithm's steps are given in the procedure 1.

Procedure 1 K-means algorithm.

```
1: Select K points as initial centroids.
2: repeat
3:     Form K clusters by assigning each point to its closest centroid.
4:     Recompute centroid of each cluster.
5: until Convergence i.e., centroids do not change.
```

1.1.2 Hierarchical Clustering

Another fundamental paradigm in cluster analysis is hierarchical clustering, in which a clustering solution is represented as a tree describing hierarchical relationships between nested clusters [3]. Many applications can benefit from the nested feature of clusters in a hierarchical domain. Clusters may have subclusters with different densities inside which create a nested structure. So hierarchical clustering can give us an insight of clusters with nested structure. In addition, tree representation of clusters fits some underlying application domains as they have nested or hierarchical structure in essence. Biological taxonomy and document categorization are examples of such a domain [4]. This type of clustering provides more information about the cluster's structures of the data than those provide by the ”flat” models. Hierarchical clustering may have more advantages than flat clustering in a way that it demonstrates data from multiple levels allowing to explore the solution from different possible perspectives. It provides a global picture of the cluster structure where each level in the hierarchy reveals some specific details about the data set. There are two basic approaches for generating a hierarchical clustering:

- Agglomerative: This approach is also known as ”bottom up” approach; objects are regarded as individual clusters in the first step and in the next steps, the closest pair of clusters are merged.
- Divisive: This approach is also known as ”top down” approach; all objects are in all-inclusive cluster and in each step, one split is performed recursively as one moves down the hierarchy.

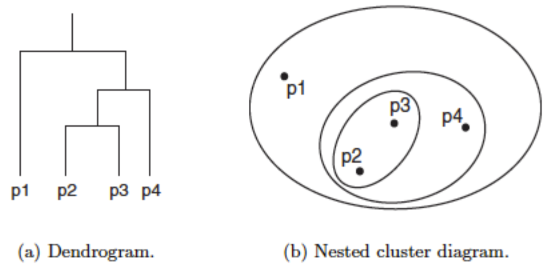


Figure 1.2: A hierarchical clustering of four points shown as a dendogram and as the nested clusters.

A hierarchical clustering is often displayed graphically using a tree-like diagram called a **dendogram**, which displays both the cluster-subcluster (parent node- child node) relationships and the order in which the clusters were merged. Figure 1.2 shows the dendogram and its nested cluster diagram.

1.1.3 Density-Based Clustering

In a density-based clustering, a cluster is a set of data objects with high density in the data space separated from other clusters by low density areas. Data objects located in low-density regions are typically considered as noise or outliers. As a consequence in density-based clusterings, clusters may have arbitrary and non-linear shapes. Density-Based Spatial Clustering of Applications with Noise (**DBSCAN**) [5] is the most widely used density based algorithm. The basic density concepts and definitions are as follows (taken from[5]):

Definition 2. (*Eps-neighborhood of a point) The Eps neighborhood of a point* p*, denoted by* $N_\varepsilon(p)$*, is defined by*$N_\varepsilon(p) = \{q \in D | dist(p, q) \leq Eps\}$.

Definition 3. ($Directly\ density - reachable$) A point p is directly density-reachable from a point q wrt. Eps, MinPts if

1. $p \in N_{Eps}(q)$.

2. $|N_{Eps}(q)| \geq MinPts$ (core point condition).

Definition 4. ($Density - reachable$) A point p is density reachable from a point q wrt. Eps and MinPts if there is a chain of points $p_1, ..., p_n, p_1 = q, p_n = p$ such that $p_i + 1$ is directly density-reachable from p_i.

Definition 5. $(Density - connected)$ A point p is density connected to a point q wrt. Eps and MinPts if there is a point o such that both, p and q are density-reachable from o wrt. Eps and MinPts. [5]

Definition 6. $(Cluster)$ Let D be a database of points. A cluster C wrt. Eps and MinPts is a non-empty subset of D satisfying the following conditions:

1. $\forall p, q$: if $p \in C$ and q is density-reachable from p wrt. Eps and MinPts, then $q \in C$. (Maximality)

2. $\forall p, q \in C$: p is density-connected to q wrt. Eps and MinPts. (Connectivity)

Definition 7. $(Noise)$ Let $C_1, C_2, ..., C_k$ be the clusters of the database D wrt. parameters ϵ_i and $MinPts_i$, $i = 1, ..., k$. Then we define the noise as the set of points in the database D not belonging to any cluster C_i , i.e. noise $= \{p \in D | \forall i : p \notin C_i\}$.

The pseudocode of DBSCAN algorithm is shown in the procedure 2:

Procedure 2 DBSCAN algorithm.

```
1: Determine the type of all points either core, border, or noise points.
2: Separate noise points from all other points and remove them.
3: Find maximal subsets of core points that are within Eps-Neighborhood of
   each other.
4: Form maximal subset of connected core points into one cluster.
5: Add each border point to the closest cluster of its associated core points.
```

1.2 Semi-Supervised Clustering

Recently, semi-supervised clustering has been of particular interest. The reason is that in many cases a small amount of knowledge is available as a prior knowledge about the data. This prior knowledge is provided by user's feedback about specific memberships or relationships of some limited number of objects. This information mostly comes in two formats: either pairwise (Must-link or Cannot-link) constraints between data items or class labels for a subset of items. A portion of the data mining literature has been dedicated to study semi-supervised clustering since prior knowledge can guide or adjust the clustering process. It should also be mentioned that for semi-supervised clustering the available knowledge is assumed to be too far from being representative of a target classification of the items, so that supervised learning is not possible [6].

Unlike traditional clustering, the semi-supervised clustering methods have a short history and fewer methods haves been published until now. Recent works in the clustering domain integrate user's knowledge with most of the well-known clustering techniques that were presented earlier in this chapter. These methods differ in the way they use prior information (pairwise constraints or subset of labeled objects): either by adapting the similarity measure or by modifying the search for appropriate clusters. We will discuss these methods further in Chapter 2.

1.3 Finding Parameters for Clustering Algorithms

The appropriate choice of the clustering algorithm and parameter settings (density threshold in the density-based clustering or the number of expected clusters) varies from one individual data set to another. The goal of the applications of clusterings may also guide the choice of the clustering algorithm and the parameter settings. The choice of the algorithm and the parameter values is not an automatic task and is typically difficult in many applications. For instance, the K-means algorithm is the most popular and efficient clustering algorithm for clusters with spherical shapes in the input space. However, many data sets in practice do not have this shape which makes it difficult to find the appropriate value for k in advance. The number of clusters k in partitional methods has a big impact on the clustering quality. Clustering quality is a term used to show to what extent the membership labels obtained by the clustering conform to the natural clustering structure of the data. Several density-based clustering algorithms require two basic parameters ε and *MinPts* as inputs. The ε parameter comes from a continuous range of values which makes it difficult to search for the suitable value. Increasing the value of *Minpts* parameter (for constant ε) will result in a larger number of data points being labeled as outliers, whereas decreasing the threshold ε (for constant *Minpts*) will have a similar effect. There are more examples of clustering algorithms that require parameter settings in order to partition objects into clusters. Manually setting parameters is difficult and does not guarantee finding the "best" parameter. The "best" clustering parameter presents a clustering partition that conforms the most with respect to the ground truth clusters. Some works suggest selecting the parameter automatically by running their clustering algorithm repeatedly for a number of values of that parameter and selecting the one which provides the "best" performance as evaluated by some internal [1] validation index.

[1]Measures the goodness of the clustering structure regardless of external information usually by considering cluster cohesion and cluster separation.

1.4 Thesis contributions

The importance of the clustering parameter selection motivates us to design a framework to find the “best” parameter for semi-supervised clusterings. Our framework denoted as “Cross-validation framework for finding clustering parameter” (CVCP) finds the “best” parameter using a limited number of labeled objects or constraints.

1.5 Dissertation outline

The rest of the dissertation is organized as follows. In Chapter 2 we talk about the problem of parameter tuning in clustering methods and introduce some major works done in unsupervised and semi-supervised scenarios. Also we will discuss approaches available to evaluate semi-supervised clusterings. In Chapter 3 we introduce our ideas to design a framework to select the appropriate parameter for semi-supervised clustering algorithms. Then we talk about how the framework is evaluated, present the results of our experiments and compare it with other methods in Chapter 4. Finally we conclude the this work in Chapter 5.

Chapter 2

Background and Related Work

The problem of determining the "best" number of clusters or finding an accurate partitioning of a data set attracts many interests in the data mining literature. In this chapter, we will discuss some of the papers presenting methods to determine the "best" number of clusters e.g., k for K-means based clusterings [2] or ε and MinPts for density based clusterings. Most of the works that are mentioned here are efforts to find the "best" parameter for unsupervised clusterings as there is a very few papers that determine the "best" parameter for semi-supervised clusterings [7]. Most published papers propose a procedure for estimating the "best" value of a parameter for a clustering method and experimentally compare it with some other methods. The first section of this chapter is dedicated to the review of papers that presented methods for estimating the "best" k for K-Means in the published literature. The second section will discuss the methods proposed for finding the"best" parameters for density based clustering methods. In the third section we will discuss existing approaches for evaluation of semi-supervised clusterings and point out their drawbacks in some of them. The last section of this chapter talks about two semi-supervised clustering algorithms that we used in our experiments.

2.1 K-means Clustering

According to [8], among different proposals in the literature for finding the "best" k, we can find the following categories in a broad sense:

- Variance-based approach: model based criterion in which the "best" k receives the highest value.
- Structural approach: involves minimizing the within-cluster cohesion while maximizing between-cluster separation for the "best" k.

- Hierarchical approach: the "best" k will be extracted from a tree obtained from a divisive or agglomerative clustering method.
- Resampling approach: choosing the "best" k by resampling methods like subsampling, bootstrapping or randomly perturbing.

In the following section we will discuss some of the significant papers from each of the above mentioned categories. We will describe their main idea and motivations along with a description of their approaches.

2.1.1 Variance-based approach

Given a data set, K-means clustering methods generally try to minimize the sum of within-cluster distances to centroids:

$$W_K(S, C) = \sum_{k=1}^{K} \sum_{i \in S_k} d(i, c_k) \tag{2.1}$$

where $C = c_1, c_2, ..., c_K$ is a set of centroids of corresponding K partitions $S = S_1, S_2, ...,$ S_K. Each point in the data set I belongs to one of the S_i cluster and d denotes any distance measure. There have been several methods in the literature to cluster a data based on the minimum of 2.1 at a specified K (W_K). Tibshirani et al. [9] in 2001 proposed the Gaps-statistics method to estimate k based on the equation 2.1. This method has become popular in the bioinformatics community. The method can use the output of any clustering algorithm in order to estimate k. The method compares the W_K with its expected value under the uniform distribution. The method assumes the data is clustered into K clusters $S_1, S_2, ..., S_K$ with S_r denoting the points that are members of cluster r, and $n_r = |S_r|$. The sum of pairwise distances for all points in cluster r is simply:

$$D_r = \sum_{i,i' \in S_r} d_{i,i'} \tag{2.2}$$

Using the equation 2.2, the objective function in the equation 2.1 is changed to the following format:

$$W_K = \sum_{r=1}^{K} \frac{1}{2n_r} D_r \tag{2.3}$$

The method compares the $\log W_K$ to its expected value under a null reference distribution of the data. The gap value associated with a cluster number K is defined as:

$$Gap_n(K) = E_n^* \log(W_K) - \log(W_K) \tag{2.4}$$

The Authors estimate that for the right value of K, the difference between the expected reference curve and the $\log W_K$ reaches its maximum value. $E_n^* \log (W_K)$ denotes the expected value based on several samples of size n from the reference distribution. So the value $\log W_K$ and its expected value are computed for all Ks in the predefined range of K using each sample of size n. The $Gap_n(K)$ for each K are also computed based on the equation 2.4. The method is evaluated using data sets generated from 5 different scenarios. The scenarios are: single cluster data in 10 dimensions, three clusters in two dimensions, four clusters in 10 and 3 dimensions and two elongated clusters in 3 dimensions. The results of Gap statistics for estimating the "best" K is compared with different indices such as "Silhouettes" [10] and "CH" [11]. "CH" is an index proposed by Calinski and Harabasz (1974), the index is defined as follows:

$$CH(k) = \frac{B(k)/(k-1)}{W(k)/(n-k)} \tag{2.5}$$

where $B(k)$ and $W(k)$ are between- and within-cluster sums of squares, with k clusters. $CH(k)$ reaches its maximum at the "best" k [11]. The "Silhouettes" index will be explained in detail in the next section. The Gap statistics clearly outperforms "Silhouettes" and "CH" in two realizations of the mentioned scenarios; finding a single cluster in the first scenario and two elongated clusters in the last scenario where "Silhouettes" and "CH" are not able to find the "best" value of k. In other scenarios, all methods perform quite well.

2.1.2 Structural approach

A number of methods introduced indices measuring within-cluster distances versus between-cluster distances. The greater the difference of these distances is the better the model fits data. One of the well-balanced coefficients that was proposed in 1990 by Rousseeuw et al. [10] is **Silhouette width**. The concept of the silhouette width focuses on the difference between the within-cluster tightness and between-cluster separation. Consider a point i in the data set X and a partition P which is the output of running a clustering algorithm on data set X. The dissimilarity value of i to all other points in the same cluster which i belongs to is denoted by $a(i)$. The dissimilarity measure can be a distance function. This value can be interpreted as how well i is matched to the cluster it is assigned to. The smaller the value , the better is the matching. The average dissimilarity of the point i to all other clusters in P is computed. The lowest average dissimilarity of i to any such cluster is denoted by $b(i)$. The value of $b(i)$ denotes the "neighboring" cluster or the cluster that is the best next cluster for point i. The silhouette width of i is defined as:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))} \tag{2.6}$$

The silhouette width values range from $[-1, 1]$. The strength of the silhouette lies in the interpretation and validation of cluster analysis results. We can consider three cases in terms of the relation of natural clusters in the data set and the partitions that are obtained by a clustering algorithm for a value of k. In the first case, the data set consists of some dense clusters which are far away from each other but the value for k is set too low. The returned partition will have some natural clusters combined in order to reduce the total number of clusters to be equal to the specified value of k. The combination of the different clusters will lead to large "within" dissimilarities resulting in a large $a(i)$ and a small $s(i)$ values for all of the objects. The second case involves setting the value of k too high which results in splitting some natural clusters in order to extract the specified number of clusters. In this scenario, the distance between points that belong to the same cluster in the natural clustering, which are now split into two or more clusters are very small on average. Hence the "between clusters" dissimilarities $b(i)$ will become very small, resulting in a small $s(i)$ values. It can be implied that based on the silhouette reasoning, the "natural" value of k gets the best value of silhouette width. The average silhouette width for each cluster is defined by the average of silhouette widths of all points belonging to that cluster. This value helps to distinguish "clear cut" from "weak" clusters. A larger average silhouette width of a cluster denotes a well-partitioned group of points. The overall average silhouette width for the entire partitions is the average of silhouette widths of clusters. The overall silhouette width helps to choose the "best" k for the data set. Each value of k will yield a different overall average silhouette width. The appropriate value of k is the one that gets the large possible value of silhouette width. The method was applied to the Ruspini data [12]. This data set is composed of 75 two dimensional points, and was originally used by Ruspini to illustrate fuzzy clustering techniques. The points are naturally from four clusters and the silhouette plots indicates that a partition into $k = 4$ clusters is probably the best. The overall average silhouette width $S(k)$ obtained is the largest for $k = 4$.

2.1.3 Hierarchical approach

A number of approaches utilize the hierarchy of clustering solutions to estimate the "best" k. Pelleg et al. [13] presented a new K-means based algorithm, X-mean, that incorporates model selection. The computations involve making local decisions about the number of centroids and which subset of current centroids should be split to produce a better and closer fit to the natural clusters. The decision to split the centroids is based on the Bayesian Information Criterion (BIC). The X-mean algorithm takes a range of k values as an input (R) and outputs a set of centroids associated with the value of k that achieves the best score by the model selection criterion. The algorithm starts clustering with the minimum value of k in R and continues to add centroids if they improve the BIC criterion until the upper bound of R is reached. During this process, the centroids set that obtains the best score with respect to BIC is the one that will be returned as the final partition by X-means. Each iteration of X-means has two steps using a value of k from the given range R as an input. The total number of iterations X-means completes is $|R|$. In each iteration i, the first step is running the conventional K-means with $K = r_i, r_i \in R$ until convergence. A stable partition will be returned as the output of the first step. The partition is passed to the second step. In the second step, an operation to find out if and where new centroids should appear will take place. The operation considers the boundaries of the regions “owned” by each centroid and splits the region of each centroid into two children (subregions) by selecting two new centroids. The new centroids are placed a distance proportional to the size of the

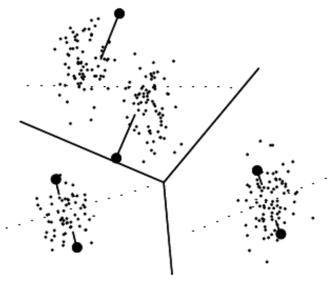


Figure 2.1: The first step of parallel local 2-means. Two new centroids are randomly selected in each subregion.

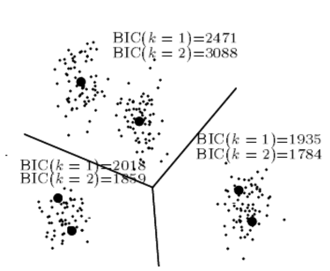


Figure 2.2: Results after parallel 2-means have terminated.

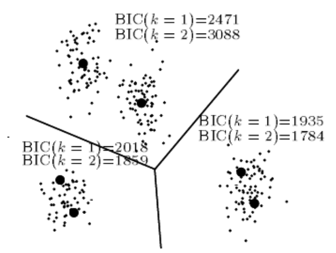


Figure 2.3: BIC is computed in each subregion for $k = 1$ and $k = 2$.

region in opposite directions along a randomly chosen vector as shown in figure 2.1 [13]. Next, in each parent region a local K-means is performed with $k = 2$ using new centroids as initial centroids. Now, the model selection criterion is evaluated here in order to test whether original parent model or the two children structure fits the real distribution better. According to the value of BIC either the parent or its offspring are killed [13], figure 2.3. The procedure continues until the upper bound for k in R is reached and the centroid setting with maximum BIC will be selected as the closest structure to the natural distribution and the best possible value for k is the number of clusters in the centroid setting [13]. This paper used the posterior probabilities $Pr[M_j|D]$ to score models where D is the data and M_j is the family of alternative models. Models are assumed to be spherical Gaussian. Experimental results on both synthetic and real life data show it is performing faster and better than K-means. Experiments were conducted using Gaussian data sets were generated as in Pelleg and Moore [14] and also on a big dataset of over 330,000 galaxies. The results show that K-means tends to over-estimate the number of classes, while X-means usually under-estimates the true K.

2.1.4 Resampling approach

Another approach toward finding the right number of clusters is resampling. Resampling is the use of many randomly produced "copies" of the data for assessing statistical properties of a utilized method. Methods for generating the random copies can fall into one of the following categories according to [8]:

(a) random sub-sampling in the data set.

(b) random splitting the data set into "training" and "testing" subsets.

(c) bootstrapping, that is randomly sampling object with replacement.

(d) adding random noise to the data objects

All methods are based on the idea that for the "best" number of k, we should expect more similar results from different copies. Here we will review one of the works in this domain by Dudoit and Fridland (2002) [15]. They present the "Clest framework" in which the number of clusters in a data set is estimated based on a prediction-based resampling. The approach that is presented in the paper is particularly focused on solving tumor classification. An important statistical problem in classifying tumor data set is determining new classes of tumor by means of gene-expression profiles. So the task of determining the right number of clusters, "k", is discussed in this paper. Authors believe that validation of clustering results can be improved by focusing on prediction accuracy. The method is provided a maximum value of k (M). The method performs the following steps for each value of k in the range $2 \leq k \leq M$.

1. Repeat the following B times:

   (a) The original set is randomly divided into two non-overlapping sets, a learning set L^b, and a test set T^b.

(b) For each value of k, the “PAM” (partitioning by medoids) clustering algorithm [16] is applied to the leaning set L^b to obtain a partition $P(.; L^b)$.

(c) A classifier $C(.; L^b)$ is built using the learning set L^b and its partition $P(.; L^b)$.

(d) Classifier $C(.; L^b)$ is applied the to test set T^b.

(e) The PAM clustering algorithm is also applied to the test set T^b and a partition $P(.; T^b)$ is obtained.

(f) $P(.; L^b)$ is compared to $P(.; T^b)$ using the Fmeasure external validation index [17].

2. The median of $|B|$ F-measure score is reported as the observed similarity statistic for k.

3. The expected observed similarity statistic for k is produced by generating a B_0 reference data sets under a null hypothesis and applying the steps 1 and 2.

4. P-value for the expected observed similarity statistic and observed similarity statistic is computed.

The estimated number of clusters $\hat{k}$ is then defined to be the k in the range $2 \leq k \leq M$ that has the largest significant difference from the expected observed similarity statistic. Clest was compared to methods such as “Gap” [9] and “silhouette” [10]. The Clest method was also evaluated using gene-expression data from four cancer microarray studies. The methods Clest and silhouette estimated the natural number of classes for all except the “NCI60” data set. It is shown that “Clest” gives uniformly good results over different range of data sets and is the most robust and accurate, compared to the other methods, but it has a weak performance when there is overlapping clusters in data sets.

2.2 Density-based Clustering

In the DBSCAN papaer [5], the author proposed a heuristic to determine the values of ε and MinPts. Later, other papers proposed improved versions of DBSCAN or different approaches for density-based clustering, which more or less followed the idea addressed in DBSCAN to set parameters. Several papers usually assign fixed values to either ε or MinPts or both.

2.2.1 The DBSCAN heuristic to determine density parameters

DBSCAN [5] is a clustering algorithm DBSCAN relying on a density-based notion of clusters. It also introduces a heuristic to estimate the value of ε and $MinPts$. This method is intended to find clusters of arbitrary shapes. The method assigns a density value to each object by counting the number of neighbor objects in ε neighborhood around the object. The objects with higher density than a threshold ($MinPts$) are considered as core objects and clusters are groups of connected core points. DBSCAN assists the user in finding the appropriate value of MinPts and ε. The method determines the *k-nearest neighbors* for each object and sorts them in a descending order to generate the *sorted k-dist graph.* The heuristic suggests that the first ”valley” of the sorted k-dist graphs is the threshold point which separate the noise objects from the objects assigned to the clusters. The *k-dist* value of the threshold point is set to be ε and the MinPts will be equal to the k. The paper states that for k larger than 4, sorted k-dist graphs are not significantly different from 4-dist graphs so in order to avoid computational costs, they consider MinPts = 4 in all experiments. The value for ε is estimated based on the first ”valley” in the 4-dist graph. While the heuristic offered by [5] is simple, it has few drawbacks. For example, finding the first ”valley” in the 4-dist graph for a data set is not always an easy automated task and it requires a human participant to determine the global parameter ε. There are other papers in the literature that propose a different density-based clustering but following the heuristics addressed in the [5].

2.3 Semi-supervised clustering algorithms

Several semi-supervised clustering algorithms have been proposed in different clustering paradigms. These works integrate previous knowledge with a clustering method from partitional, hierarchical and density-based paradigms. Also, these methods differ in the way they use the prior information (pairwise constraints or subset of labeled objects): either by adapting the similarity measure or by modifying the search for appropriate clusters. In this section, we will introduce some semi-supervised partitional clustering and semi-supervised density-based clustering methods. We evaluate the performance of our framework in finding the “best” parameters using methods from partitional and density-based clustering paradigms. **MPCKMeansS** [18] and **FOSC** [19] are methods we used inside our framework as semi-supervised Kmeans-based and semi-supervised density-based clustering algorithms, respectively. Both methods take set of constraints in form of Must-link/Cannot-link as prior knowledge. These algorithms will be described in this section.

2.3.1 Semi-supervised partitional clustering algorithms

In this section, we explain two semi-supervised K-means based clustering papers. First, we discuss MPCKmeans that is presented by Bilenko et al. [18] in 2004. Second, we discuss the Seeded K-means and Constrained-Kmeans presented by Basu et al. [20] in 2002.

MPCKmeans [18] is a K-means based clustering method that integrates two approaches. The paper introduces a method that incorporates constraints into the objective function and uses them to learn a distance metric. Since K-means is not able to incorporate pairwise constraints directly, the authors reformulate the basic objective function of the K-means by adding terms to penalize constraint violations. The objective function is given in the equation 2.7 where X is a set of data points, M and C are sets of Must-link and Cannot-link constraints. $\mathbb{1}$ is the indicator function, $\mathbb{1}[true] = 1$ and $\mathbb{1}[false] = 0$.

$$\mathcal{J}_{pckmeans} = \sum_{x_i \in X} \|x_i - \mu_i\|^2 + \sum_{(x_i,x_j)\in M} w_{ij}\mathbb{1}\left[l_i \neq l_j\right] + \sum_{(x_i,x_j)\in C} \bar{w}_{ij}\mathbb{1}\left[l_i = l_j\right] \quad (2.7)$$

Constraints can be employed to learn a distance metric. The learning method tries to find a new metric that minimize the distances between same-cluster objects. The Euclidean distance is parameterized by a positive-definite matrix A as follows:

$$\|x_i - \mu_i\|_A = \sqrt{(x_i - \mu_i)^T A (x_i - \mu_i)} \quad (2.8)$$

Instead of using one metric weight for all clusters, in this method each cluster has its own weight matrix denoted by A_h for cluster h and the objective function is updated using the weight matrices as follows. :

$$\mathcal{J}_{pckmeans} = \sum_{x_i \in \mathcal{X}} \left(\|x_i - \mu_i\|^2_{A_{l_i}} - \log\det(A_{l_i})\right) \quad (2.9)$$

Combining the two equations 2.7 and 2.9 leads to the following objective function that minimizes cluster dispersion under the learned metrics, while reducing constraint violations. f_M and f_C are penalty factors for violating Must-link and Cannot-link constraints, respectively. f_M gives a higher penalty to two points that are farther away from each other than to nearby points which are involved in a Must-link constraint. f_C gives a higher penalty to two nearby points than to distant points which are involved in a Cannot-link. So the combined objective function of this algorithm is as follows:

$$
\begin{aligned}
\mathcal{J}_{pckmeans} = & \sum_{x_i \in \mathcal{X}} (\|x_i - \mu_i\|^2_{A_{l_i}} - \log \det(A_{l_i})) \\
& + \sum_{(x_i, x_j) \in M} w_{ij} f_M(x_i, x_j) \mathbb{1}\left[l_i \neq l_j\right] \\
& + \sum_{(x_i, x_j) \in C} \bar{w}_{ij} f_C(x_i, x_j) \mathbb{1}\left[l_i = l_j\right]
\end{aligned}
\tag{2.10}
$$

The MPCKMeans algorithm starts by initializing clusters. The points are assigned to the clusters based on the objective function 2.10. Point o is assigned to the cluster l such that l minimize 2.10 for o. After all points are assigned to the clusters, the centroids of clusters are updated. The metric is updated based on new centroids and cost of violations. This process will be repeated until convergence.

Basu et al. [20] present a semi-supervised clustering method with seeding. The paper introduces two variants of semi-supervised K-means. The seed set is the subset of labeled objects provided by a user. The method assumes that there is an object from each class in the seed set forming l subpartitions where l is the number of classes in the data set. The first variant $Seeded - Kmeans$ benefits from the seed set in the initialization phase of K-means. Instead of randomly choosing initial centroids, the algorithm takes the mean of each subpartition of the seed set as initial centroids. The rest of the algorithm is the same as the K-means. The second variant is $Constrained - Kmeans$ which uses $Seeded - Kmeans$ in the initialization step. In this step, clusters of objects are updated and objects are reassigned iteratively. Only objects not in the seed set are reassigned and the cluster labels of seed objects are kept unchanged.

2.3.2 Semi-supervised density-based clustering algorithms

In this section, we explain two semi-supervised density-based methods. $\mathbf{C-DBSCAN}$ is proposed by Ruiz et al. in 2007 [21] and $\mathbf{FOSC}$ is proposed by Campello et al. in 2013 [19].

$\mathbf{C-DBSCAN}$ [21] is an extension of DBSCAN for semi-supervised clustering using Must-Link and Cannot-link constraints. The algorithm partitions the data space with a kd-tree. C-DBSCAN creates “local clusters” while enforcing Cannot-link constraints. All leaf nodes in the tree are traversed: if data points in the same leaf are involved in a Cannot-link constraint, then each point in the leaf becomes a singleton cluster. If there are no points in the leaf that are involved in a Cannot-link constraint then the DBSCAN labels all unlabeled points in the leaf of the tree as either noise or core points. If the number of points in a ε-

distance of a point in the leaf is less than MinPts then the point is noise; otherwise all the points that are density-reachable from this point form a “local cluster”. Finally the clusters are created by merging local clusters under Must-link constraints.

Campello et al. presented FOSC [19] in 2013 to extract clusters from hierarchies optimally. Usually flat clusters are extracted from an appropriate level of a hierarchy through a horizontal cut. The appropriate level in a hierarchy can be found either manually or automatically. Sometimes it is not possible to extract the right solution through a single horizontal cut according to the nested nature of a data set. FOSC is a framework for extracting clusters from a hierarchy, corresponding to local cuts at different levels of the hierarchy. The authors utilize the idea of local cuts instead of a single global cut to extract the optimal clusters. Applying local cuts in a hierarchy is similar to using different density thresholds for different subsets of a data set to perform the clustering. Prior knowledge in the format of constraints will help to find local cuts that either globally maximize constraints satisfaction or minimize constraints violations in the hierarchy. The set of eligible clusters that can be in the flat solution are all clusters (nodes) in the hierarchy except for the root. The method also assumes that each object belongs to a single cluster. The flat solution also cannot contain both a child node and its parent. To enforce this condition in maximizing the number of satisfied constraints, an objective function is formulated as follows:

$$J = \frac{1}{2n_c} \sum_{j=2}^{\kappa} \gamma(x_j) \tag{2.11}$$

$$\underset{\delta_1,\delta_2,\ldots,\delta_\kappa}{max} \; J$$
$$\begin{cases} \delta \in \{0,1\} & i = 2, ..., \kappa \\ \sum_{j \in I_h} \delta_j = 1, & \forall \text{h such that } C_h \text{ is a leaf cluster,} \end{cases} \tag{2.12}$$

where $\delta i(i = 2, ..., \kappa)$ is an indicator that denotes whether cluster C_i is included in the flat solution $(\delta i = 1)$ or not $(\delta i = 0)$. I_h is set of indices for clusters in a path from the root (excluded) to the cluster $C_h(included)$ [19] and $\gamma(x_j)$ is the number of constraints involving object x_j that are satisfied (not violated). Based on the above equations FOSC returns a set of clusters from different level of hierarchy as a flat solution.

2.4 Semi-supervised clustering evaluations

In this section we will discuss the approaches available in the literature to evaluate the result of semi-supervised clustering algorithms. There are different methods in the literature that employ constraints or a set of labeled objects in semi-supervised clusterings. These methods report the impact of using constraints or labeled objects for semi-supervised clusterings in their evaluation. Some of these methods include training data (constraints or labeled objects) when evaluating clustering performances and reports only overall accuracy. The C-DBSCAN [21] method, that was discussed earlier, considers different data sets with known clusters and the authors report Rand Index [22] over entire data set considering the constraints in the evaluation. In these type of methods, the training and test sets are not completely independent.

There are methods that exclude constraints or labeled instances in their test set and report clustering accuracy only for unconstrained instances. Wagstaff et al. [23] proposed a constrained K-means clustering in 2001. They modify K-means [2] in order to use prior knowledge to boost clustering quality. The main difference between this algorithm and K-means is in the step of assigning objects to their closest clusters. The algorithm will not assign an object o to a cluster if it violates the specified constraints. In terms of evaluation, the authors report clustering performance including and excluding constraints in their validation, presenting two sets of numbers: the overall accuracy for the entire data set, and accuracy on a held-out test set using Rand Index [22]. In the latter case, they perform a 10 fold cross-validation where 9 folds are taken to generate constraints and the performance is only computed on the 10th fold.

The $Seeded - Kmeans$ method [20] was explained in the previous section. For the evaluation, the authors compare the performance of these methods on two data sets (CMU 20 Newsgroups data and Yahoo! News data) with various seeding and noise levels, using 10-fold cross validation. In each run, they take 90 percent of the data set as the training set that produces the seed set and the remaining 10 percent of data set is used as the test set. The clustering quality measured by mutual information (MI) [24] is reported as an average over different runs. Mutual information is a symmetric measure for determining the similarity of the cluster labels and class labels.

In the latter case, the methods report accuracy scores that are based on evaluation on only the test sets which share no common information with training sets. In the next chapter, we present our framework to select an appropriate clustering parameter. In the second version of framework, the training and test sets are completely independent for evaluations. We report clustering performance using the framework suggested parameter value on test sets only.

Chapter 3

Parameter Selection for Semi-Supervised Clustering Algorithms

In the previous chapters, we discussed the importance of choosing an appropriate clustering parameter and a clustering method. We also discussed some of the papers that focus on finding the “best” parameter values for clustering algorithms (e.g., k in Kmeans-based clustering methods). Few number of papers present heuristics to determine the values of $MinPts$ and ε for density-based clustering methods. Most of these works study the problem of parameter selection in an unsupervised clustering domain. These papers select the “best” parameter value based on different internal measures. Traditional clustering algorithms can not take advantage of constraints on cluster membership of some points or a limited number of labeled points even when it does exist. However, there are cases in which the experimenter may be able to provide some background knowledge; such as instance labels or pairwise constraints that could be useful for clustering a data. As we discussed in chapter 2, only very few papers propose a parameter selection method for semi-supervised clustering algorithms. Here, we introduce a framework to select the “best” parameter value for semi-supervised clustering algorithms. The framework can be applied to any semi-supervised clustering algorithm in order to find the“best” value of the clustering parameter.

Before we get to the framework structure and steps, there are some symbols needed to be explained:

- P: A set of instance labels provided as a background knowledge.
- C: A set of constraints to use as a background knowledge.
- α: A clustering parameter, $MinPts$ or k.

- S: A range of values for the parameter α.

3.1 Framework overview

Given a range of values for a clustering parameter, our method, denoted as "cross-validation framework for finding clustering parameters" ($CVCP$) finds the "best" value of the clustering parameter based on limited number of constraints or labeled points provided by a user. A semi-supervised clustering method achieves the highest possible clustering quality using the "best" value of the clustering parameter. The best clustering quality represents a grouping of the data set that conforms most with the constraints or class labels given as input to a semi-supervised clustering algorithm. In this section, we explain our motivations to propose CVCP and how we use background knowledge to develop this framework.

3.1.1 Motivation

Consider a semi-supervised clustering method which takes as input a value of α and prior knowledge in the form of partially labeled points P or a set of constraints C provided by a user. CVCP is a cross-validation framework that evaluates a clustering algorithm's performance for a given value of the clustering parameter based on the constraints satisfaction score computed with respect to the given labeled points or constraints. For a fixed set of constraints C (the set of labeled points P), CVCP computes the constraints satisfaction score for each value s in the range S and the parameter value $s' \in S$ associated with the highest constraints satisfaction score is selected.

Also, CVCP can be used to gain insight into which clustering paradigm is most suitable for a data set X. For example, we can use CVCP to find the "best" value of k for a semi-supervised K-means clustering method on the data set X and record the clustering quality (q_k) associated with the "best" value of k. Then, we can apply CVCP to find the "best" value of $MinPts$ for a semi-supervised density-based clustering method and compare its best clustering quality (q_{MinPts}) with q_k. Since we are comparing the performances of two different clustering methods at their "best" value of clustering parameters, we can decide which clustering method is the better match for the data set X.

3.1.2 Problem definition

Definition 8. Given a data set X with a small subset of labeled points P or a set of constraints C and a range of values S for parameter α, find a parameter value in the parameter set S that gives the best possible semi-supervised clustering solution on X using the available background knowledge given to the semi-supervised clustering method.

CVCP's inputs includes a data set X, a form of prior knowledge P or C and a set of parameter values S. In the next section, we introduce the first version of CVCP. In the first version of the CVCP, the format of prior knowledge could be a subset of labeled points or a set of constraints. This version will be modified later and the reasons will be explained in the next sections. The form of the background knowledge that can be used in the second version of CVCP is a subset of labeled points P provided by the user. It should be noted that the information given by the pairwise constraints is weaker than that given by a subset of labeled data. Class labels can generate pairwise constraints, but the inverse is not true.

3.2 CVCP I

In this section, we explain the structure of CVCP. Figure 3.4 shows the diagram of the first version of CVCP. As we discussed, the general idea is to use cross-validation to evaluate a clustering algorithm performance in satisfying the limited number of constraints/labeled points using a parameter value and then selecting the "best" parameter value upon the cross-validation framework.

3.2.1 Step I: Assigning constraints to classes

CVCP performs a preprocessing step on the provided partially labeled points or the set of constraints. In this step, if the input is a subset of labeled points then the complete graph of constraints is generated from the subset of labeled points. For every two labeled points in this subset, one Must-Link constraint is generated if they have the same cluster labels otherwise a Cannot-link constraint is generated. If a set of constraints is provided then the transitive closure of all available constraints is computed.

After generating constraints as mentioned above, we treat the constraints as objects (entities), create two classes and assign each object in the graph to one of these classes. Each object is assigned to class label ML or CL based on its type (Must-link and Cannot-link) respectively (figure 3.1).

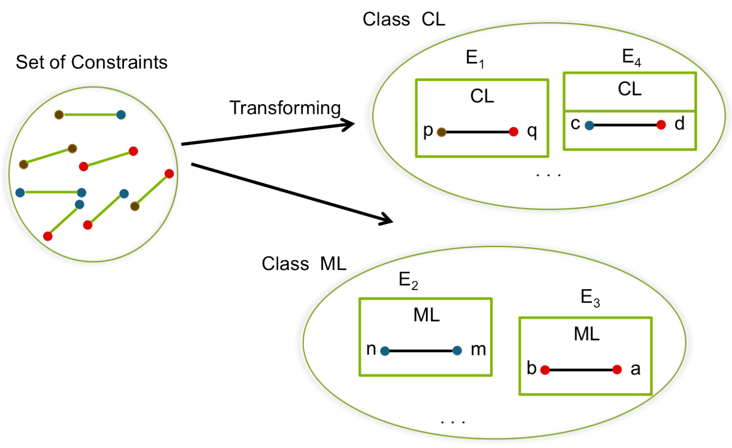


Figure 3.1: Treating constraints as new objects or entities

As we explained, CVCP evaluates the clustering quality for the output partition of a semi-supervised clustering algorithm that is obtained with a value of the clustering parameter, based on the constraints satisfaction with respect to a limited number of labeled points or constraints. To measure the constraints satisfaction, we compute the predicted label for each binary relation from the graph. For each binary relation, we check if the points involved in the binary relation with class label ML (CL) have the same (different) cluster labels in the output partition then the predicted class label for this binary relation is ML (CL) otherwise the predicted class label is CL (ML) (figure 3.2).

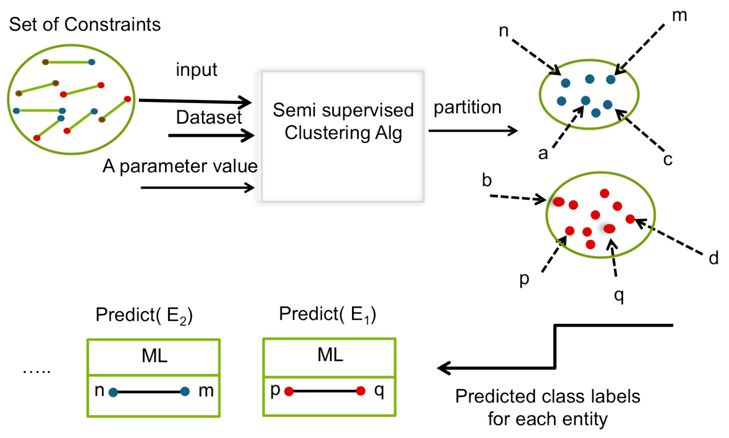


Figure 3.2: Compute predicted class labels for each entity

Now, we can use a classification measure to count the number of correct and incorrect predicted labels for all binary relations in the graph and report a classification score (constraints satisfaction score) (figure 3.3).

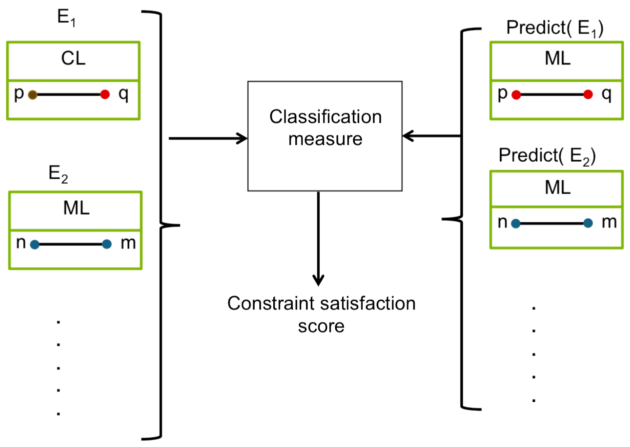


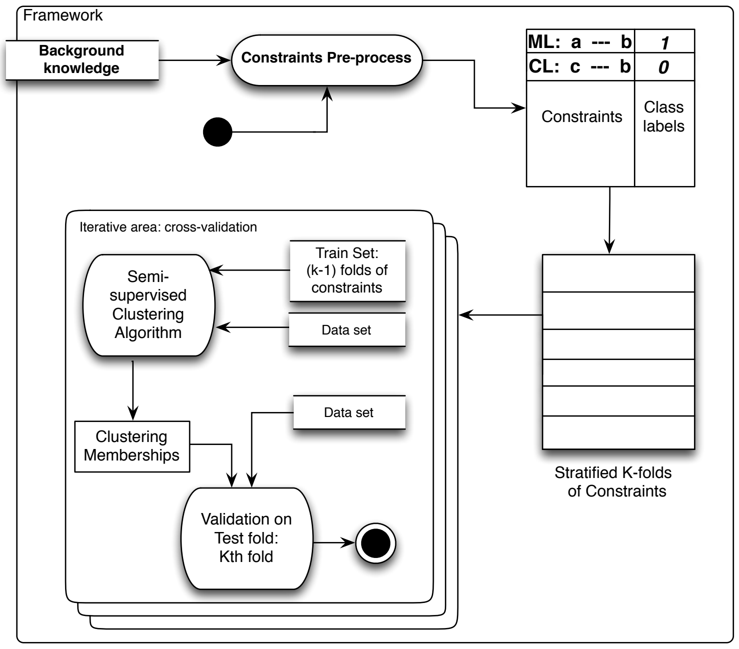
Figure 3.3: Computing classification score

So the clustering evaluation with respect to a limited number of labeled points or constraints is transformed to a classification evaluation. To sum up, the first step of CVCP includes a preprocessing phase to generate the constraints denoted as G_C and assigning them to classes ML or CL which represents objects with type of Must-link and Cannot-link, respectively.

3.2.2 Step II: Stratified k-fold cross-validation,

In this step, we perform the stratified K-fold cross-validation. The binary relations from two classes ML and CL are partitioned across the K non-overlapping folds. In the stratified K-fold cross-validation, each fold contains approximately the same proportion of two types of the class labels. It means that the proportion of Must-link and Cannot-link constraints in each fold is about the same of their original proportion in G_C. Of K folds, one fold is retained as the test data and the other $K-1$ folds are used as the training data. Cross-validation consist of repeating this process K times, with each fold is used once as the test data. In each iteration i, the semi-supervised clustering algorithm takes the data set X , a parameter value s and the training sets as inputs. Then the quality of constraints satisfaction with respect to the test fold is computed, denoted as q_{xi}. The final quality of the clustering method denoted as q_x using parameter value s over the constraint set G_C, is the average value of q_{xi}, $1 \leq i \leq K$.

The next section will point out a conceptual drawback of the first version of CVCP and the second version of CVCP will be explained.



Framework
Background knowledge
Constraints Pre-process
ML: a --- b
1
CL: c --- b
0
Constraints
Class labels
Iterative area: cross-validation
Semi-supervised Clustering Algorithm
Train Set: (k-1) folds of constraints
Data set
Clustering Memberships
Data set
Validation on Test fold: Kth fold
Stratified K-folds of Constraints

Figure 3.4: CVCP -I Diagram

3.3 Conceptual Drawback

The approach in which we partition the constraints in G_C into K-folds is problematic. Following this approach may result in having some folds with "overlapping" constraints. Some of the constraints that are implicitly derived from the training folds can be found explicitly in the test fold. consider the two following scenarios:

Scenario 1 Let a, b, c be 3 labeled points from the set P. Assume a and b have the same labels and the Must-Link constraint generated by two points a and b is in the ith fold (without loss of generality). Let furthermore assume that c and b have different class labels and the Cannot-link generated by c and b is in the ith fold and the Cannot-link constraint generated using c and a is in the jth fold.

Scenario 2 Let a, b, c be 3 labeled points from the set P. Assume a and b have the same class labels and the Must-Link constraint generated by two points a and b is in the ith fold (without loss of generality). Let furthermore assume that c and b have the same class labels and the Must-link generated using c and b is in the ith fold and the Must-link constraint generated using c and a is in the jth fold .

What happen in the scenario 1 and 2, if the semi-supervised clustering algorithm uses the ith fold as a training set and uses the jth fold as a test set? The independency of of the training set and the test set is violated and the clustering algorithm is evaluated on binary relations in the test set that some of them can be obtained by transitive closure of the binary relations in the training set. So, overfitting to the model that is learned may happen in these scenarios. We will present a solution to overcome this issue in the next section.

3.4 CVCP II

In this section, we explain the structure of the second version of the CVCP framework. The framework takes a subset P of labeled points along with one data set X and a parameter s as inputs. Figure 3.8 shows the diagram of the second version of CVCP. We also explain the solution to the conceptual drawback discussed in the previous section. The main difference between the current version and the first version is that the current version takes prior knowledge only in the form of a subset of labeled points. The remaining structure of CVCP including cross-validation are still the same as in the first version.

3.4.1 Step I: Partition of the set of labeled points

We first equally distribute labeled points in the set P into K folds and then the process of generating constraints will be followed in the next step (figure 3.5).

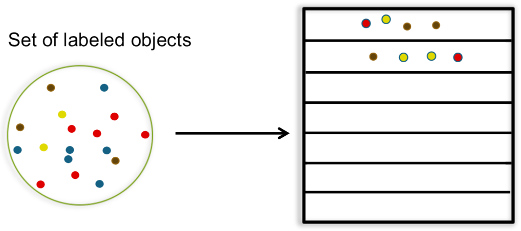


Figure 3.5: Partition of the set of labeled points

3.4.2 Step II: Producing constraints

In the second step, we take the $K-1$ folds as the training set and collect all the points from these folds to generate constraints. We leave the Kth fold as the test fold and generate constraints from the points in this fold. The constraints are treated as objects with class labels ML or CL the same as in the step I from the first version of CVCP. This version of CVCP partitions labeled points across the folds first and then generates constraints from them. This approach guarantees that folds do not share constraints transitively. So, no overfitting happens and the second version of CVCP becomes more robust (figure 3.6).

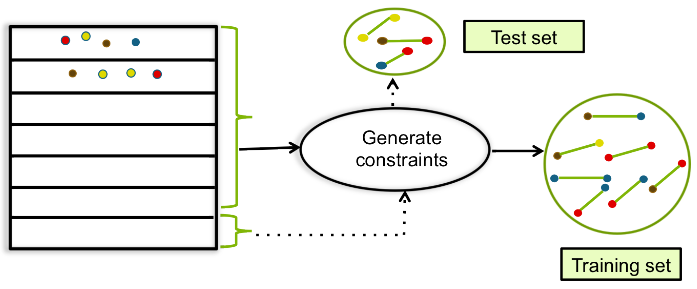


Figure 3.6: Producing constraints

3.4.3 Step III: Cross validation

In this step, K-fold cross-validation is performed in the same way as in the first version of CVCP. The training and test sets in each iteration of the cross-validation is created as we explained in the previous step. Then the quality of the clustering is evaluated by computing constraints satisfaction score using a classification measure as in the step II of the first version of CVCP. The purpose of using cross-validation is to obtain a robust estimation of how well the semi-supervised clustering method satisfies the constraint set (figure 3.7).

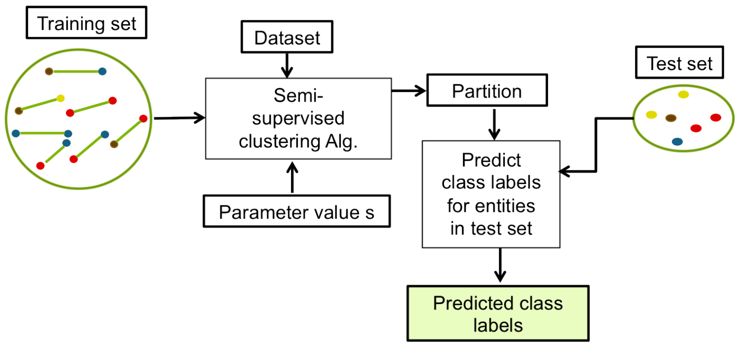

Training set
Dataset
Semi-supervised clustering Alg.
Parameter value s
Partition
Test set
Predict class labels for entities in test set
Predicted class labels

Figure 3.7: Computing predicted labels

Figure 3.8 shows all the above steps in one diagram.

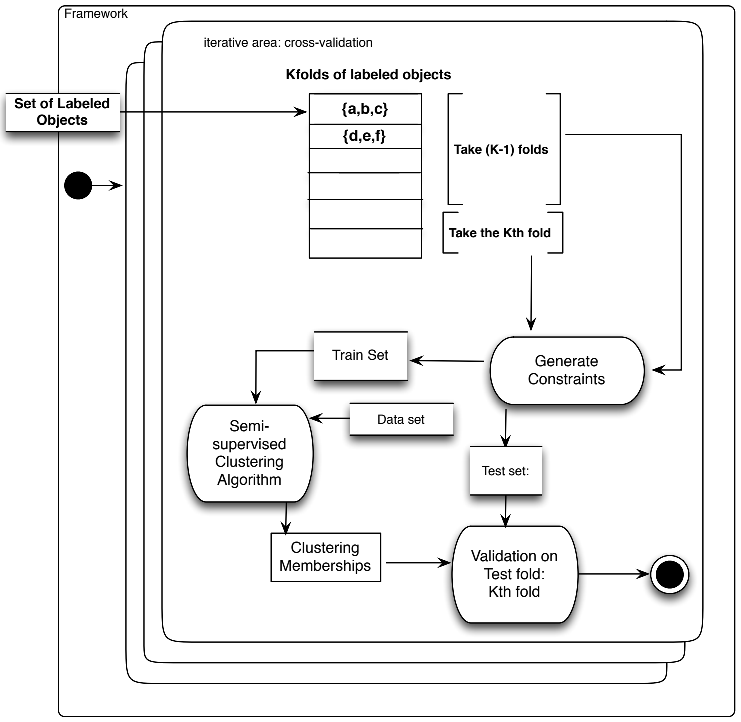


Figure 3.8: CVCP -II Diagram

3.5 Parameter Selection based on CVCP

We explained the structure and steps of CVCP in the previous section. Here, we explain how CVCP will help the user to select the "best" parameter value that achieves the best possible clustering solution.

The algorithm to select the "best" parameter by CVCP can be found in procedure 3.

Procedure 3 CVCP application for finding the "best" parameter value.

```
Input:Data set $X$, a fixed set of labeled points $P$, a range of parameter values $S$.
for $s \in S$ do
    Run CVCP on data set $X$ using set $P$ with parameter value $s$.
    Record classification score for $s$.
end for
Find the highest classification score and its associated parameter $s'$.
Output:Parameter $s'$ is the "best" parameter value suggested by CVCP.
```

CVCP classification scores will be recorded for every value of the parameter α in the range S and the parameter value associated with the highest classification score is returned as the "best" value for the parameter α (figure 3.9).

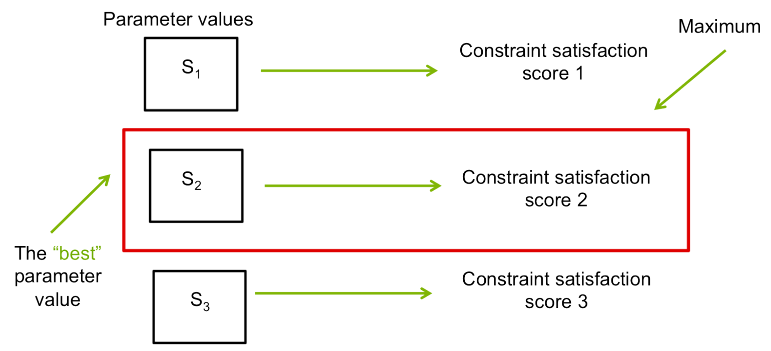

Parameter values
Maximum
S_1
Constraint satisfaction score 1
S_2
Constraint satisfaction score 2
The "best" parameter value
S_3
Constraint satisfaction score 3

Figure 3.9: Parameter selection process

We expect that the "best" parameter value suggested by CVCP can be generalized to the entire data set so that using this parameter value achieves the highest clustering quality for the entire data set. In the next chapter, we will present experiments designed to evaluate the CVCP performance in selecting the "best" parameter for semi-supervised clustering methods. We will also present experiments that show the correlation of the semi-supervised clustering performances evaluated using the ground truth and the CVCP internal classification scores for different values in S.

Chapter 4

Experiments

In the previous chapter, we explained the structure of the CVCP framework to find the "best" clustering parameter values for semi-supervised clustering algorithms. As we explained, the "best" parameter value for a semi-supervised clustering method is selected based on the highest constraints satisfaction score with respects to the constraints derived from the given labeled points. We design the evaluation methods in order to identify the usefulness of CVCP in finding the "best" parameters for semi-supervised clustering algorithms. Our evaluation method is to run the semi-supervised clustering method using the CVCP selected parameter and then evaluate how well the output clustering partition conforms the ground truth. The semi-supervised clustering performance obtained using the CVCP selected parameter is compared to the clustering performances obtained using other existing methods that find "best" parameters. We also compare the clustering performance obtained using the CVCP selected parameter to the expected performance when the parameter has to be guessed.

This chapter introduces details of data sets and the evaluation methods along with results according to each evaluation method. It worth to mention again that we evaluate the CVCP performance for finding the "best" value of MinPts and k for the semi-supervised clustering methods $\mathbf{FOSC}$ [19] and $\mathbf{MPCKmeans}$ [18], respectively. Both methods take a set of Must-link/Cannot-link constraints as input. Before we get to the evaluation methods and results, there are some symbols needed to be explained:

- P: A set of instance labels provided as a background knowledge.
- β: A percentage of labeled points that are involved in the constraints.
- ρ: Value of correlation coefficient.
- S: A range of values for MinPts or k.

- M: The maximum value in S.

4.1 Data sets

For the evaluation we used real data sets from different domains, image and UCI data sets as well as synthetic data sets both with a variety of characteristics (no. of objects, dimensionality, and no. of clusters). We use some of the class labels as input for the semi-supervised clustering method and the rest is for validation purposes only. We will present the details of these data sets in the following section.

4.1.1 ALOI Data sets

ALOI (Amsterdam Library of Object Images) [25] consists of categories of color images for one-thousand objects, recorded for scientific purposes. Each category contains one-hundred recordings of a specific object where the viewing angle, illumination angle and illumination color for that object were systematically varied. We used image sets that were created by Horta and Campello [26]. Each data set is created by randomly selecting images (without repetition) from k different categories, $k = 2, 3, 4$ and 5. An image collection is composed of hundred data sets for each k. We used the $k5$ image collection that consists of 100 data sets each having 125 objects from five categories. Each data set has 25 objects from each category. The descriptor for each objects is color moments with 144 attributes.

4.1.2 UCI Data sets

UC Irvine Machine Learning Repository [27] maintains 246 data sets as a service to the machine learning community. Among these data sets, we used the *Iris, Wine, Ecoli* and *Ionosphere* data sets. The “Iris” data set maybe is the best known database to be found in the pattern recognition literature. The data set contains 3 classes of 50 instances each with 4 attributes, where each class contains a type of iris plant and attributes for one instance are the lengths and widths of its sepal and petal. One class is linearly separable from the two which are not linearly separable from each other. The “Wine” data set contains 178 objects in 13 dimensions, with 3 classes. These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines. The “Ecoli” data set contains 336 objects in 7 dimensions, with 8 classes. The classes represents the cellular localization sites of proteins and the attributes are sets of biological properties. The “Ionosphere” data set contains classification of radar returns from the ionosphere classified into “good” and “bad” classes. “Good” radar returns have evidence of some type of structure in the ionosphere. “Bad” returns are those that do not; their signals pass through the ionosphere [27]. The data set contains 351 instances with 34 continuous attributes. Each instance in this data set, is described with 17 pulse number with 2 attributes per pulse corresponding to the values of the complex electromagnetic signals.

4.1.3 Julia Handl Data sets

Julia Handl [28] developed two data set generators in order to obtain test data of sufficient complexity for clustering. The first generator is a Gaussian cluster generator which is based on a standard cluster model using multivariate normal distributions. There are data sets with 2 and 10 dimensions and the number of clusters are 4, 10, 20 and 40. For each of the 8 combinations of a cluster number and a dimension, 10 different data sets were generated, giving 80 data sets in all. The second generator creates ellipsoidal clusters with the major axis at an arbitrary orientation. There are data sets with 50 and 100 dimensions and the number of clusters are 4, 10, 20 and 40. For each of the 8 combinations of a cluster number and a dimension, 10 different data sets were generated, giving 80 data sets in all. We used 10 data sets in 2 dimensions with 4 clusters and 10 data sets in 10 dimensions with 4 clusters created by the Gaussian cluster generator. We report the average clustering performance on these collections (2D Julia-Handl and 10D Julia-Handle).

4.2 Evaluation Measures

We used two different types of F-measure as our evaluation measures in this work. First, we describe a classification measure, denoted as Average-Fmeasure, to be used within the CVCP framework to evaluate the constraints satisfactions in order to select the “best” parameter value. Second, we report the Overall F-measure [17] to evaluate the output clustering partition of the semi-supervised clustering algorithm obtained using the “best” parameter value selected by CVCP. The Overall-Fmeasure is explained in more details in this section.

The F-Measure is an evaluation measure defined based on the harmonic mean [1] of precision and recall and is displayed as a percentage value. Precision is defined as the fraction of objects that are correctly assigned to the cluster. Recall is defined as the extent to which a cluster contains all objects of a specified class [1]. Precision, recall and F-measure are defined as follows:

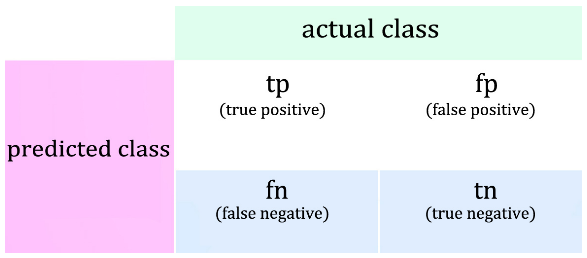
[1]The harmonic mean of $x_1, x_2, ..., x_n$ is defined as $H = \left(\frac{1}{n}\sum_{i=1}^{n} x_i^{-1}\right)^{-1}$

- $$Precision = \frac{tp}{tp + fp}$$

- $$Recall = \frac{tp}{tp + fn}$$

- $$F - measure = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

True positive, false positive and false negative are denoted by tp, fp and fn. The table 4.1 shows the above notions.



	actual class	
predicted class	tp (true positive)	fp (false positive)
	fn (false negative)	tn (true negative)

Figure 4.1: Classification Table.

Average-Fmeasure As we explained, the complete graph of constraints is produced based on a limited number of labeled points as the input to the semi-supervised clustering algorithm. We assigned the pairs of points (treated as objects) from the complete graph of constraints to classes $\{1\}$ and $\{0\}$. Classes $\{1\}$ and $\{0\}$ represent Must-link and Cannot-link pairs of points respectively. In our experiments, we derived the constraints from a randomly selected limited number of labeled points, and usually the number of Cannot-link constraints is larger than the number of Must-link constraints in the complete graph of constraints, therefore usually classes $\{1\}$ and $\{0\}$ have different cardinalities. Since the number of Cannot-link constraints in the complete graph is considerably larger than the number of Must-link constraints, it is possible that the clustering method tends to output a partition with a larger number of clusters in order to satisfy the Cannot-link constraints. Also this may prevent the CVCP framework to find the appropriate parameter value for the semi-supervised clustering method. The classification measure that is used within the CVCP framework computes the Average-Fmeasures of classes $\{1\}$ and $\{0\}$. This measure is a simple average of F-measures for the class $\{1\}$ and F-measures for the class $\{0\}$ to cancel out the imbalanced number of pairs of points that belong to classes $\{1\}$ and $\{0\}$.

We compute the F-measure for each class based on the number of correct and incorrect predicted labels for pairs of points from the complete graph. The Average-Fmeasure is formulated as follows:

- $$Fm_{C_1} = \frac{2 \times Precision_{C_1} \times Recall_{C_1}}{Precision_{C_1} + Recall_{C_1}}$$
- $$Fm_{C_0} = \frac{2 \times Precision_{C_0} \times Recall_{C_0}}{Precision_{C_0} + Recall_{C_0}}$$
- $$Average - Fmeasure = \frac{Fm_{C_0} + Fm_{C_1}}{2}$$

Overall-Fmeasure The overall-Fmeasure is commonly used to compare how similar two clustering results are by evaluating the accuracy of produced clustering partitions. We use Overall-Fmeasure [17] to evaluate the clustering results of the semi-supervised clustering algorithm obtained using the "best" parameter value selected by CVCP with respects to the ground truth. It should be mentioned that, the Overall-Fmeasure is reported on a part of data set which does not have the labeled points that are involved in the constraints.

Let assume the set of natural classes is $C = \{C_1, C_2, ..., C_m\}$ and the clustering is $K = \{K_1, K_2, ..., K_n\}$. The recall, precision and F-measure for natural class C_i and cluster K_j are as follows:

- $$Recall(C_i, K_j) = \frac{n_{ij}}{|C_{ij}|}$$
- $$Precision(C_i, K_j) = \frac{n_{ij}}{|K_{ij}|}$$
- $$F(C_i, K_j) = \frac{2 \times Recall(C_i, K_j) \times Precision(C_i, K_j)}{Recall(C_i, K_j) + Precision(C_i, K_j)}$$

where n_{ij} is the number of points from natural class C_i that belongs to the cluster K_j. The success in finding the natural class C_i can be considered as the maximum$F(C_i, K_j)$ for C_i when $1 \leq j \leq n$. The overall-Fmeasure is the weighted sum of such maximum F-measures for all natural classes which is:

$$F - measure(C) = \sum_{C_i \in C} \frac{|C_i|}{|D|} \max_{K_j \in K} F(C_i, K_j) \quad (4.1)$$

where $|D|$ is the number of all points in the data set.

4.3 Parameters

As mentioned earlier, we considered two types of semi-supervised clustering algorithms in this work. CVCP selects parameters Minpts and k for a density-based semi-supervised clustering and a semi-supervised Kmeans-based clustering, respectively.

- **MinPts,** describes the minimum number of points in the ε neighborhood of a core point. We used a range of $[3 : 3 : 24]$ for the set S. As it is often the range which is widely used in the literature.
- **k,** denotes the number of clusters to be found by K-means, which partitions n points into k clusters in which each points belongs to the cluster with the closest mean. In this work, we used the range of $[2 : 1 : M]$ for the values of k in S. M is a reasonable upper bound for the number of clusters.

4.4 Evaluation I: Correlation between CVCP internal classification scores and the semi-supervised clustering scores

We propose the CVCP framework that selects the “best” parameters to improve the expected performance of a semi-supervised clustering. CVCP checks the constraints satisfaction score with respect to a limited number of labeled objects or constraints for each value of the parameter in S and selects the parameter value that creates a partition that conforms most with the constraints or labeled objects. CVCP evaluates the constraints satisfaction using the average F-measure. The evaluation presented here investigates how well the semi-supervised clustering performances for different values of the parameter are correlated with the CVCP's internal classification scores. The pseudo code for this evaluation is shown in the procedure 4. This evaluation records the classification scores and the semi-supervised clustering scores for each value $s \in S$ and computes their correlation. A high correlation indicates that CVCP approximates the clustering performance for different parameter values well and can improve the expected performance. The correlation coefficient is computed as follows:

$$\rho_{X,Y} = corr(X,Y) = \frac{cov(X,Y)}{\sigma_X \sigma_Y} = \frac{E\left[(X-\mu_X)E(Y-\mu_Y)\right]}{\sigma_X \sigma_Y} \tag{4.2}$$

where $cov(X,Y)$ denotes the covariance of data X and Y, μ and σ denote the mean and standard deviation of a data.

Procedure 4 Experiment: Correlation of the semi-supervised clustering performances and CVCP internal classification scores

```
Input: Data set $X$, a fixed set of labeled points $P$ and a range of the parameter
    values $S$.
1: for $s \in S$ do
2:     Record the constraints satisfaction score for constraints derived from
       $P$, computed within the CVCP framework for the parameter value $s$.
       (Classification score)
3:     Run the semi-supervised clustering method with $s$.
4:     Record clustering quality of the output partition. (Clustering score)
5: end for
6: Compute correlation coefficient of the classification and clustering scores.
```

4.4.1 Evaluation I using FOSC

This section presents the average correlation values obtained by running the experiment in the procedure 4 for FOSC. The average results over 50 experiments are reported for different data sets and different numbers of constraints. The number of constraints is reported in the form of percentage of labeled points that are involved in the constraints, $\beta = 2\%, 5\%, 10\%$ and 20%. For data set collections such as ALOI and Julia-Handl, the average results of all data sets in the collections are reported, respectively. We depict the curves of the clustering and classification scores for one data set from ALOI and for one data set from Julia-Handl collections as representatives examples.

ALOI

Figure 4.2 depicts the curves for the FOSC clustering scores and the constraints satisfaction scores obtained within the CVCP framework for the data set $k5-59$ from ALOI. The correlation coefficient in this case is $\rho = 0.9989$. Table 4.1 shows the average correlation values for 100 data sets from the ALOI collection for different number of constraints.

$\%$ labeled points used as input	Correlation coefficient
5	0.8019
10	0.9674
20	0.9687

Table 4.1: Average correlation values of 50 experiments on ALOI

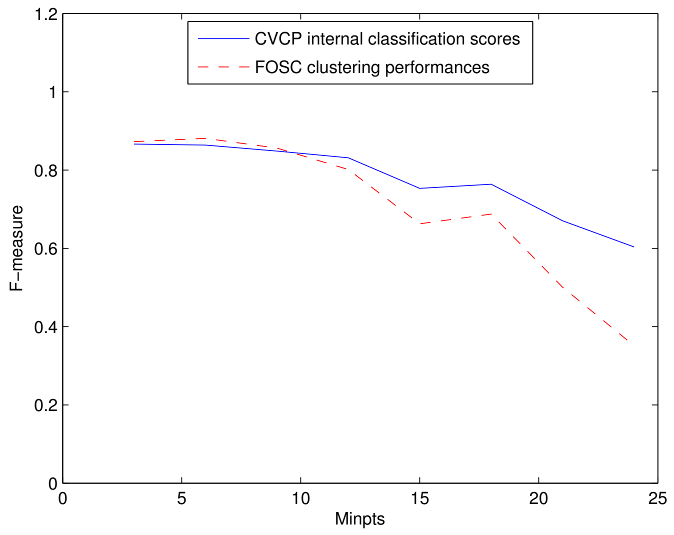


Figure 4.2: Data set $k5-59$ from ALOI collection, $\beta = 10\%$

The above figure and table show that FOSC performance for different values of MinPts is highly correlated with the CVCP classification scores.

Julia-Handl

Figures 4.3 and 4.3 depict the FOSC clustering scores and the constraints satisfaction scores obtained within the CVCP framework for the data sets no.5 and no.8 from the 2D Julia-Handl collection and 10D Julia-Handl collection, respectively. In these cases, the correlation coefficients are $\rho = 0.9794$ and $\rho = 0.6959$, respectively. Tables 4.2 show the average correlation values for 10 data sets from the 2D Julia-Handl collection for different number of constraints. Table 4.3 show the average correlation values for 10 data sets from the 10D Julia-Handl collection for different number of constraints.

$\%$ labeled points used as input	Correlation coefficient
2	0.7884
5	0.7920
10	0.8096

Table 4.2: Average correlation values of 50 experiments on 2D Julia-Handl

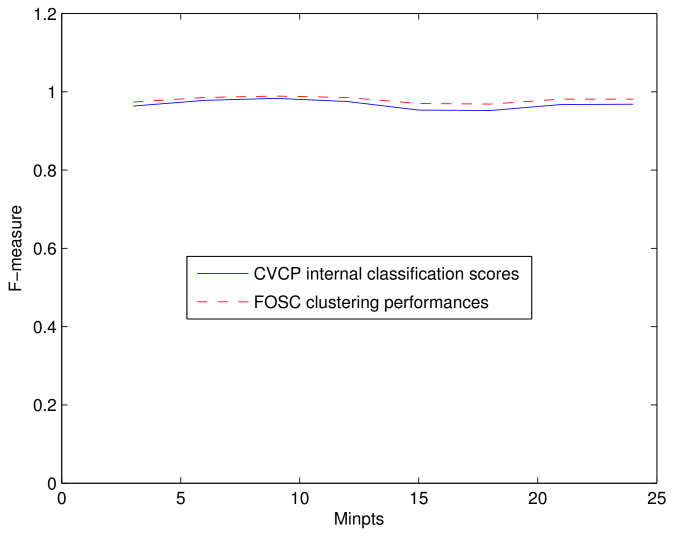


Figure 4.3: Data set no.5 from 2D Julia-Handl collection, $\beta = 5\%$

% labeled points used as input	Correlation coefficient
2	0.7803
5	0.8739
10	0.9260

Table 4.3: Average correlation values of 50 experiments on 10D Julia-Handl

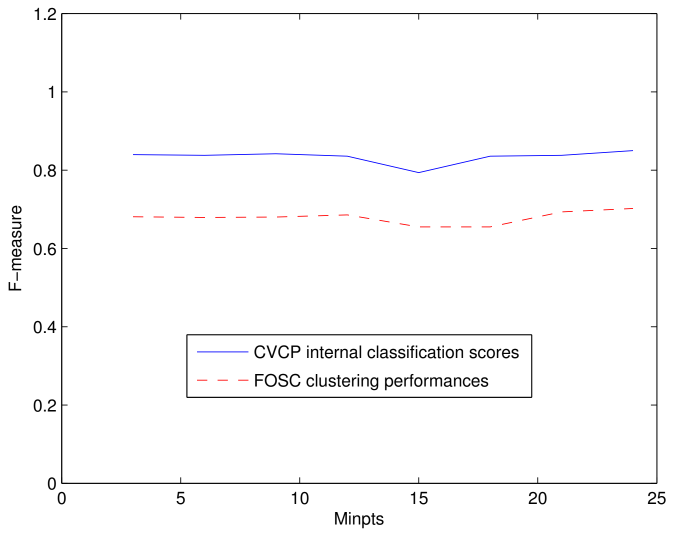


Figure 4.4: Data set no.8 from 10D Julia-Handl collection, $\beta = 2\%$

A strong correlation is obtained for the data set no.5 from the 2D Julia-Handl collection as depicted in figure 4.3. Figure 4.3 shows that FOSC is insensitive to the choice of MinPts value for the data set no.5 and has a good clustering performance for each value of the MinPts. The correlation values increase when the number of given constraints are increased for these data sets.

UCI

This section presents tables of the average correlation values for the data sets Iris, Wine, Ionosphere and Ecoli from the UCI repository. The figures depict the curves of the FOSC clustering scores and the constraints satisfaction scores obtained within the CVCP framework for each data set for different number of constraints. The figures and tables show that FOSC clustering scores and CVCP classification scores have strong correlations generally.

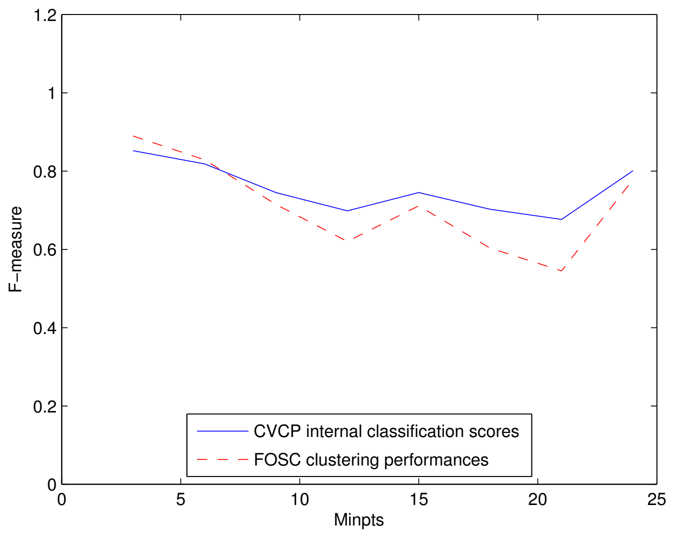


Figure 4.5: Data set Iris, $\beta = 20\%$

% labeled points used as input	Correlation coefficient
5	0.6818
10	0.6125
20	0.9902

Table 4.4: Average correlation values of 50 experiments on Iris

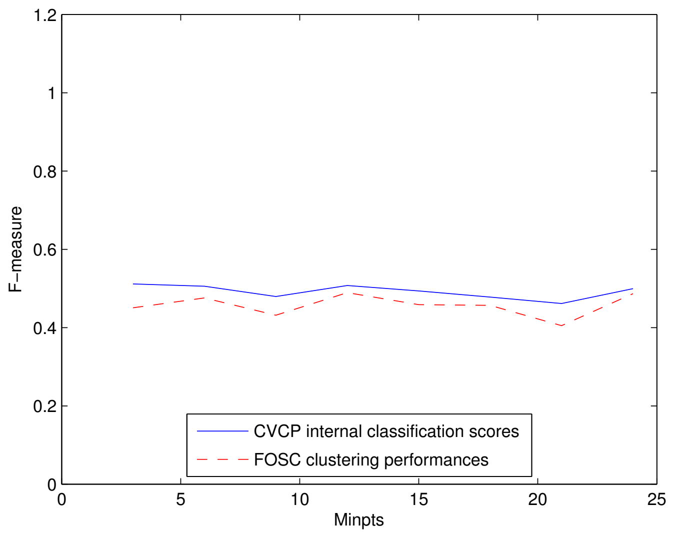


Figure 4.6: Data set Wine, $\beta = 20\%$

% labeled points used as input	Correlation coefficient
5	0.9020
10	0.7880
20	0.9381

Table 4.5: Average correlation values of 50 experiments on Wine

% labeled points used as input	Correlation coefficient
5	0.9177
10	0.9888
20	0.9695

Table 4.6: Average correlation values of 50 experiments on Ionosphere

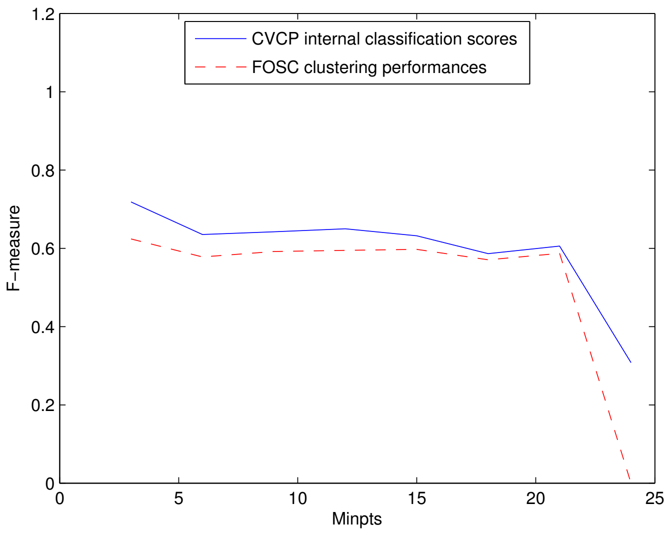


Figure 4.7: Data set Ionosphere, $\beta = 20\%$

% labeled points used as input	Correlation coefficient
5	0.6880
10	0.8819
20	0.4570

Table 4.7: Average correlation values of 50 experiments on Ecoli

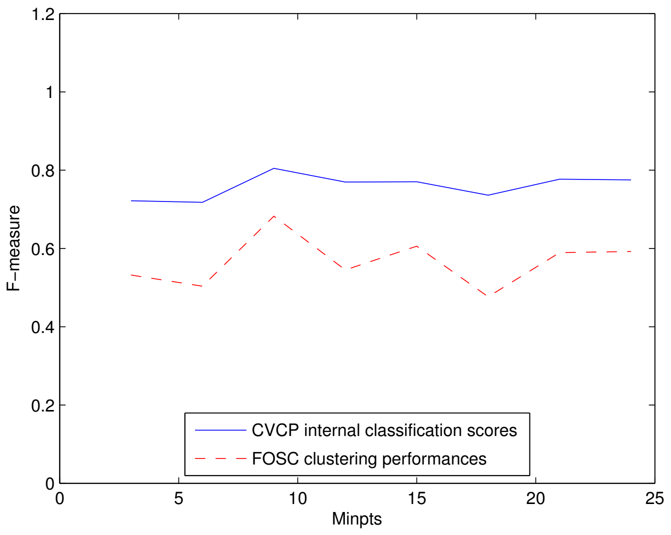


Figure 4.8: Data set Ecoli, $\beta = 10\%$

4.4.2 Evaluation I using MPCKmeans

This section presents the average correlation values obtained by running the experiment in the procedure 4 for MPCKmeans. The average results over 50 experiments are reported for different data sets and different number of constraints. The number of constraints are reported in the form of percentage of labeled points that are involved in the constraints, $\beta = 2\%, 5\%, 10\%$ and 20%. For data set collections, such as ALOI and Julia-Handl, the average results of all data sets from the collections are reported. We depict the curves of the clustering and classification scores for one data set from ALOI and for one data set from Julia-Handl collections as representatives examples.

ALOI

Figure 4.9 depicts the curves for the MPCKmeans clustering scores and the constraints satisfaction scores obtained within the CVCP framework for the data set $k5-78$. The correlation coefficient in this case is $\rho = 0.8819$. Table 4.8 shows the average correlation values for 100 data sets from the ALOI collection for different number of constraints.

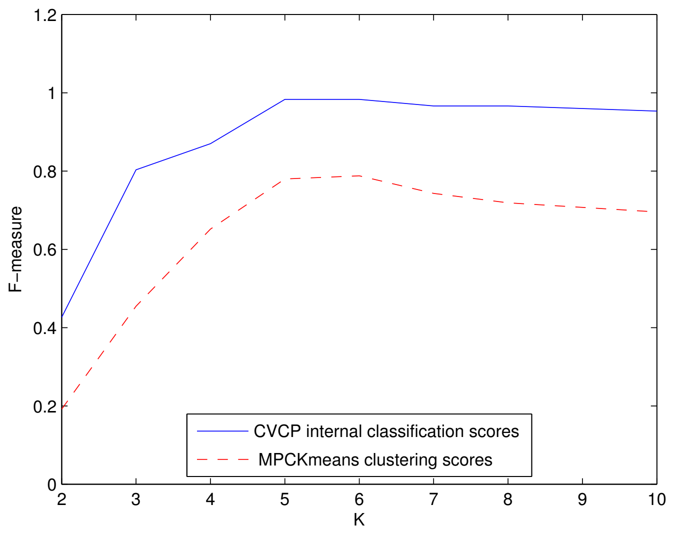


Figure 4.9: Data set $k5 - 78$ from the ALOI collection, $\beta = 5\%$

% labeled objects used as input	Correlation coefficient
5	0.9661
10	0.9237
20	0.9238

Table 4.8: Average correlation values of 50 experiments on ALOI

The results show a high correlation values for the data set $k5 - 78$ and for the ALOI collection in general. As we mentioned, the ALOI collection has 100 data sets with 5 classes each. Figure 4.9 shows that MPCKmeans has its best performance for $k = 5$ and the CVCP internal classification scores also reaches its maximum for $k = 5$. CVCP approximates well the MPCKmeans clustering performance for different values of MinPts.

Julia-Handl

Figures 4.10 and 4.11 depict the MPCKmeans clustering scores and the constraints satisfaction scores obtained within the CVCP framework for the data set no.5 from the 2D Julia-Handl collection and for the data set no.8 from the 10D Julia-Handl collection. In these case, the correlation coefficients are $\rho = 0.8261$ and $\rho = 0.4261$, respectively. Table 4.9 show the average correlation values for 10 data sets from the 2D Julia-Handl collection for different number of constraints. Table 4.10 show the average correlation values for 10 data sets in the 10D Julia-Handl collection for different number of constraints.

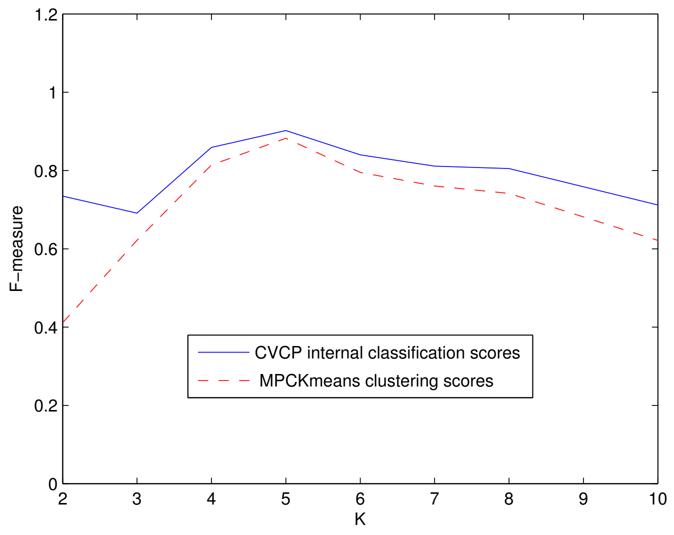


Figure 4.10: Data set no.4 from 2D Julia-Handl collection, $\beta = 5\%$

Table 4.9: Average correlation values of 50 experiments on 2D Julia-Handl

% labeled objects used as input	Correlation coefficient
2	0.7102
5	0.6433
10	0.6402

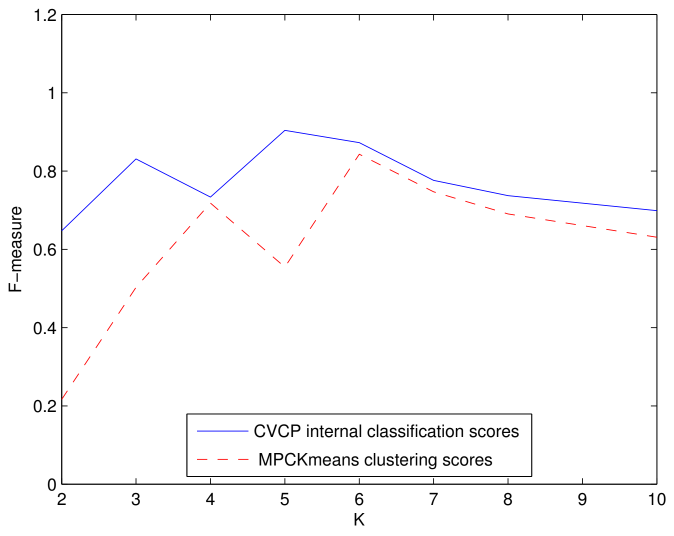


Figure 4.11: Data set no.8 from 10D Julia-Handl collection, $\beta = 5\%$

Table 4.10: Average correlation values of 50 experiments on 10D Julia-Handl

$\%$ labeled objects used as input	Correlation coefficient
2	0.5960
5	0.5964
10	0.5690

All data sets from the 2D and 10D Julia-Handl collections have 4 classes. As depicted in figure 4.11, for the data set no.8 from the 10D Julia-Handl collection, MPCKmeans has the highest clustering score for $k = 6$ and CVCP has the highest clustering score for $k = 5$. In this case, MPCKmeans is not able to find the natural clusters accurately even if it is provided with the number of natural classes.

UCI

This section presents tables of the average correlation values for the data sets Iris, Wine, Ionosphere and Ecoli from the UCI repository. The following figures depict the curves for the MPCKmeans clustering scores and the constraints satisfaction scores obtained within the CVCP framework for each data set at a different number of constraints.

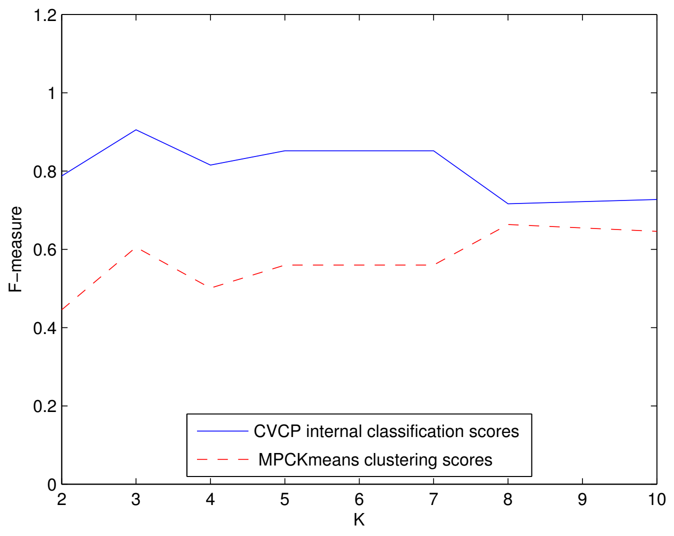


Figure 4.12: Data set Iris, $\beta = 20\%$

% labeled objects used as input	Correlation coefficient
5	-0.1643
10	0.0062
20	-0.3155

Table 4.11: Average correlation values of 50 experiments on Iris

Figure 4.12 shows that the curves are negatively correlated for the data set Iris. The Iris data set has 3 classes but the MPCKmeans performance reaches its maximum for $k = 8$. This result can give us an insight that MPCKmeans is not a suitable clustering method for the data set Iris. MPCKmeans is not able to find the ground truth clusters for the natural number of classes that is provided by the CVCP as the “best” value of k.

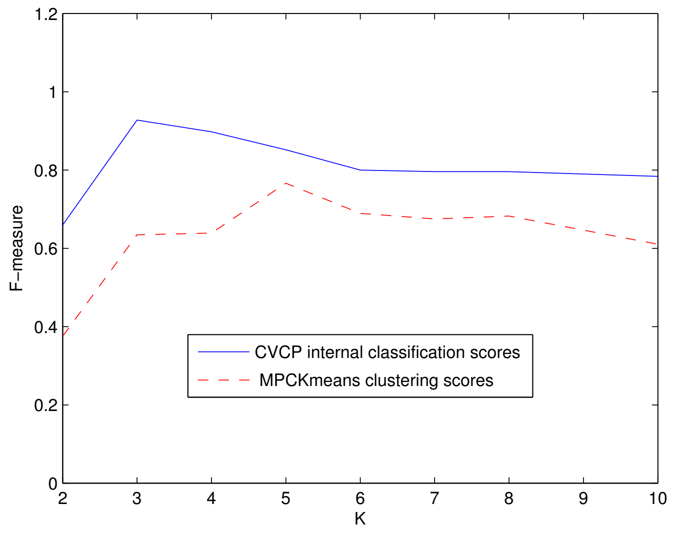


Figure 4.13: Data set Wine, $\beta = 10\%$

% labeled objects used as input	Correlation coefficient
5	0.7021
10	0.6639
20	0.2282

Table 4.12: Average correlation values of 50 experiments on Wine

Figure 4.13 shows the curves for the MPCKmeans clustering scores and the CVCP internal scores are correlated for the wine data set. In this case also, MPCKmeans is not able to find the ground truth clusters accurately for the "best" value of the parameter provided by CVCP. Wine has 3 classes and MPCKmeans reaches its maximum for $k = 5$ while the CVCP internal scores has its maximum at $k = 3$. In this case, the correlation value decrease significantly for $\beta = 20\%$. When the number of labeled points involved in the constraints increases, the imbalance between the number of Cannot-link and Must-link constraints increases. The low correlation value at $\beta = 20\%$ may relate to the fact that the more Cannot-link constraints are given to a clustering algorithm, the more the clustering algorithm tends to split the data in order to satisfy the Cannot-link constraints. This reason may lead MPCKmeans to have its maximum value of performance at larger ks when the "best" k is 3.

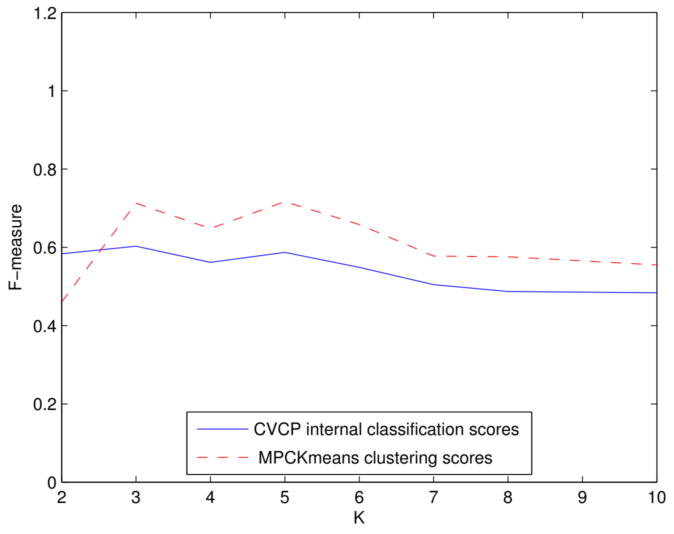


Figure 4.14: Data set Ionosphere, $\beta = 20\%$

% labeled objects used as input	Correlation coefficient
5	0.5735
10	0.4863
20	0.4211

Table 4.13: Average correlation values of 50 experiments on Ionosphere

Ionosphere has 2 classes. Figure 4.14 shows that the MPCKmeans reaches its maximum clustering score at $k = 5$ whereas the CVCP internal scores reaches its maximum at $k = 3$. MPCKmeans does not partition the Ionosphere data set well for the natural number of classes or the "best" value of k that is provided by CVCP.

% labeled objects used as input	Correlation coefficient
5	0.4360
10	-0.0508
20	0.1017

Table 4.14: Average correlation values of 50 experiments on Ecoli

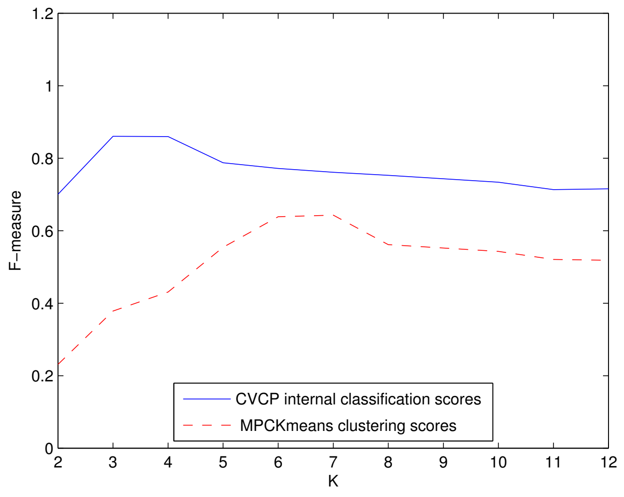


Figure 4.15: Data set Ecoli, $\beta = 10\%$

Figure 4.15 shows MPCKmeans has its maximum clustering score at $k = 7$ for the Ecoli data set , which has 8 classes. So MPCKmeans has its maximum for a value of k close to the natural number of classes while CVCP has its maximum classification score at $k = 3$. The number of points in each class in the Ecoli data set is different. Ecoli has classes with cardinalties $\{143, 77, 52, 35, 20, 5, 2, 2\}$. Figure 4.16 shows the imbalanced number of points in each class. Ecoli has three large classes, therefore more points from these classes are involved in the constraints. CVCP select the "best" value of k based on the number of satisfied constraints. We observed that the in most of the experiments the minimum number of distinct class labels is five and the large portion of the points which are involved in the constraints are from the large three clusters. This condition lead the CVCP framework to select the $k = 3$ in most of the experiments.

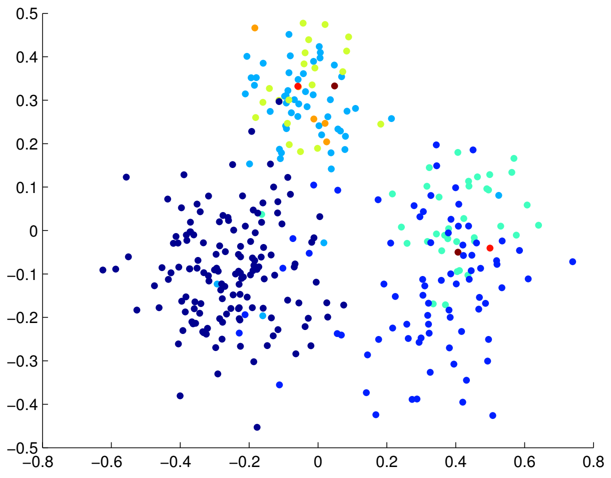


Figure 4.16: 2D scatter plot for Ecoli

4.5 Evaluation II: Clustering performance using the CVCP selected parameter

For both FOSC and MPCKmeans methods, we compared the clustering performance obtained using the CVCP selected parameter value to the expected clustering performance when the parameter value has to be guessed. In addition, the MPCKmeans performance obtained using the CVCP selected value of k are compared to the MPCKmeans performance based on the Silhouette width [10]. The Silhouette width was described in the chapter 2. The results of this evaluation using FOSC and MPCKmeans methods for different data sets are presented in section 4.5.1 and 4.5.2.

4.5.1 FOSC performance using the CVCP selected MinPts

CVCP finds the “best” MinPts as an input for FOSC. The procedure 5 compares the FOSC performance using the MinPts value selected by CVCP with the FOSC expected performance. In one experiment, CVCP takes a data set and a set of constraints derived from a set of labeled points and a range of MinPts values S as inputs and selects the “best” MinPts value from S. FOSC uses the “best” value of MinPts and returns the clustering partition. The clustering quality of this partition is the “FOSC performance using the CVCP selected parameter”. For computing the FOSC expected performance, we run FOSC for each value of MinPts in S and then average all the obtained clustering qualities. The FOSC expected performance shows the average performance of FOSC when the MinPts value have to be selected randomly. This experiment is repeated 50 times and the average of "FOSC performances using the CVCP selected parameter" and "FOSC expected performance" on a data set are reported.

Procedure 5 Experiment: FOSC performance using the CVCP selected parameter.

```
Input: Data set $X$, a fixed set of labeled objects $P$ and a range of MinPts
    values $S$.
1: Run CVCP for $X$ using constraints derived from $P$ to select the "best"
   value of MinPts, $s'$.
2: Run FOSC with $s'$: ( FOSC performance using the CVCP selected param-
   eter).
3: for $s \in S$ do
4:     Run FOSC with $s$.
5:     Record clustering quality of the output partition with respect to the
       ground truth (excluding the points in $P$).
6: end for
7: Average all recorded clustering qualities: (FOSC expected performance).
```

Results

This section reports the results of the experiment in the procedure 5 for different data sets. The results are reported for different number of constraints. The number of constraints are reported in the form of percentage of labeled objects that are involved in the constraints, $\beta = 2\%, 5\%, 10\%$ and 20%. In tables and figures of this section, "CVCP-Q" and "CVCP-std" denote the average and standard deviation of the FOSC performances using the CVCP selected MinPts over 50 experiments. "Expected-Q" and "Expected-std" denote the average and standard deviation of the FOSC expected performances over 50 experiments. For data set collections, such as ALOI and Julia-Handl the average results of all data sets in the collections are reported. We also depict the values of "FOSC performance using the CVCP selected parameter" and "FOSC expected performance" over 50 experiments for one individual data set from these collections as a representatives example.

ALOI

Figure 4.17 shows the values of "FOSC performance using the CVCP selected parameter" and "FOSC expected performance" over 50 experiments for the data set $k5 - 59$ from the ALOI collections. Each experiment uses a different set of constraints. Table 4.15 shows results for different number of constraints for the collection ALOI. The results reported in table 4.15 are averaged of 100 data sets in the ALOI collection.

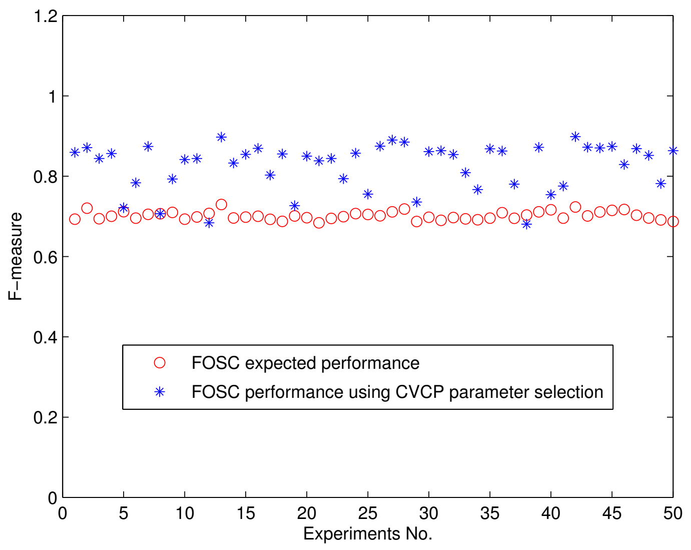


Figure 4.17: Data set $k5-59$ from ALOI, $\beta = 10\%$.

Table 4.15: Average scores for 50 experiments on ALOI

Performance / β	CVCP-Q.	Expected-Q.	CVCP-std.	Expected-std.
5	0.7489	0.7154	0.0531	0.0039
10	0.8485	0.7293	0.0620	0.0071
20	0.8569	0.7290	0.0415	0.0106

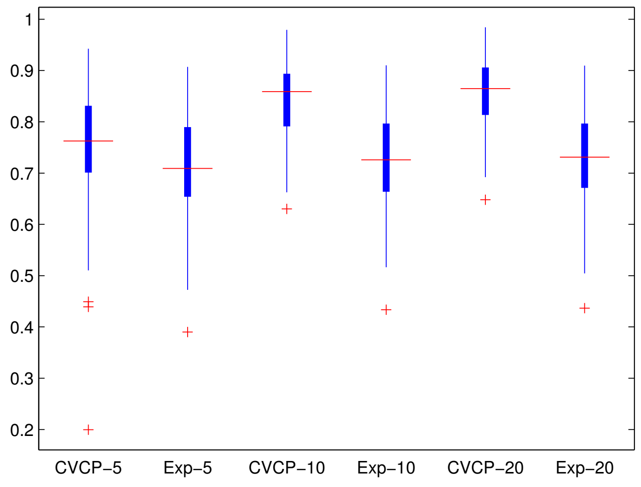


Figure 4.18: Boxplot of CVCP-Q and Expected-Q with different number of constraints - ALOI collection.

Figure 4.18 depicts the boxplot of values in table 4.15. We can see that the FOSC performance is improved if the Minpts values are selected based on the CVCP framework. Here, the average performance improves when using larger number of constraints.

Julia-Handl

Figure 4.19 shows the values of "FOSC performance using the CVCP selected parameter" and "FOSC expected performance" over 50 experiments for the data sets no.5 in the 2D Julia-Handl collection. Figure 4.21 shows the values of "FOSC performance using the CVCP selected parameter" and "FOSC expected performance" over 50 experiments for the data sets no.8 in the 10D Julia-Handl collection.

Tables 4.16 and 4.17 show the average results of 10 data sets from the 2 dimensional Julia-Handle collection and 10 data sets from the 10 dimensional Julia-Handle collection, respectively. Figures 4.20 and 4.22 depict the boxplots of values in tables 4.16 and 4.17, respectively.

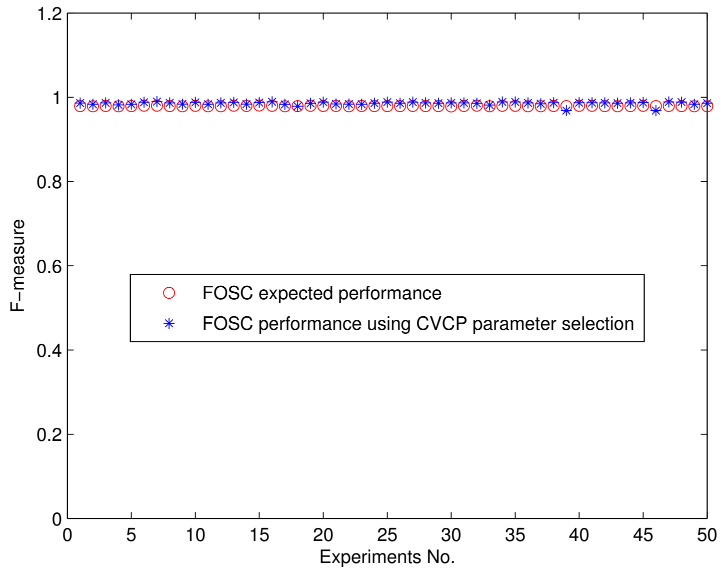


Figure 4.19: Data set no.5 from 2D Julia-Handle , $\beta = 5\%$.

Table 4.16: Average scores for 50 experiments on 2D Julia-Handle

Performance / β	CVCP-Q.	Expected-Q.	CVCP-std.	Expected-std.
2	0.9812	0.9802	0.0062	0.0004
5	0.9861	0.9844	0.0022	0.0006
10	0.9821	0.9801	0.0025	0.0007

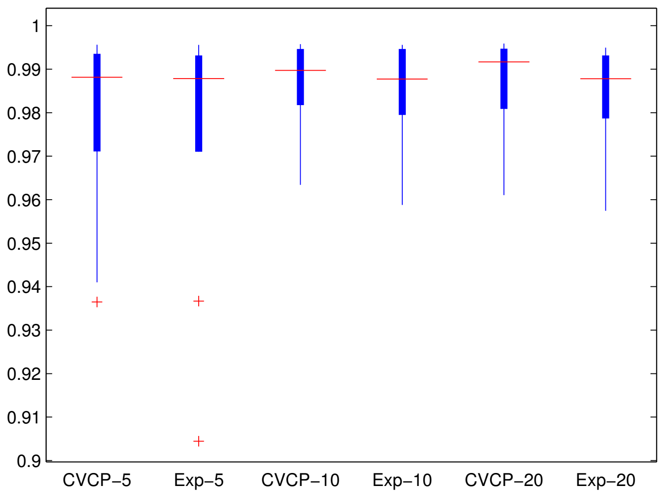


Figure 4.20: Boxplot of CVCP-Q and Expected-Q with different number of constraints - 2D Julia-Handle collection.

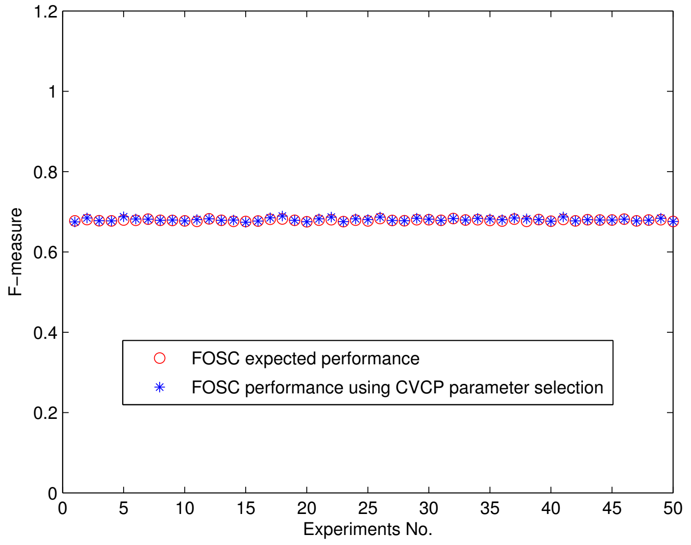


Figure 4.21: Data set no.8 from 10D Julia-Handle, $\beta = 2\%$.

Table 4.17: Average scores for 50 experiments on 10D Julia-Handle

Performance / β	CVCP-Q.	Expected-Q.	CVCP-std.	Expected-std.
2	0.8891	0.8868	0.0092	0.0009
5	0.9045	0.8914	0.0085	0.0015
10	0.9103	0.8940	0.0066	0.0017

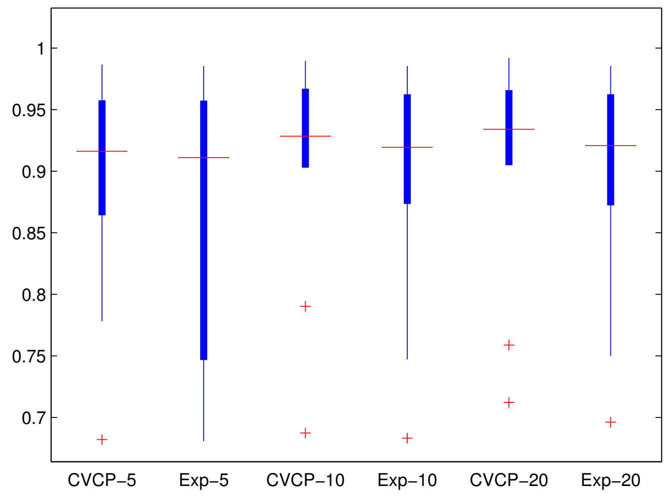


Figure 4.22: Boxplot of CVCP-Q and Expected-Q with different number of constraints - 10D Julia-Handle collection

The results for Julia-Handl collections shows that "CVCP-Q" and "Expected-Q" are very close and both have high clustering qualities. As we discussed in section 4.4.1, FOSC is not sensitive to the value of MinPts in partitioning data sets from Julia-Handl collections and presents partitions which highly conform to the ground truths for every value of MinPts in S. In this case, CVCP finds several "best" values of MinPts since FOSC partitions the data sets well enough regardless of the value of MinPts, leaving no much room for improvements.

UCI

Figures 4.23, 4.24, 4.25 and 4.26 show the values of "FOSC performance using the CVCP selected parameter" and "FOSC expected performance" over 50 experiments for the Iris, Wine,Ionosphere and Ecoli data sets from UCI repository, respectively. Tables 4.18, 4.19, 4.20 4.21 show the results for different number of constraints for these data sets.

Table 4.18: Average scores for 50 experiments on Iris

Performance / β	CVCP-Q.	Expected-Q.	CVCP-std.	Expected-std.
5	0.7251	0.6982	0.0360	0.0042
10	0.7615	0.7006	0.0401	0.0066
20	0.8251	0.7116	0.0554	0.0126

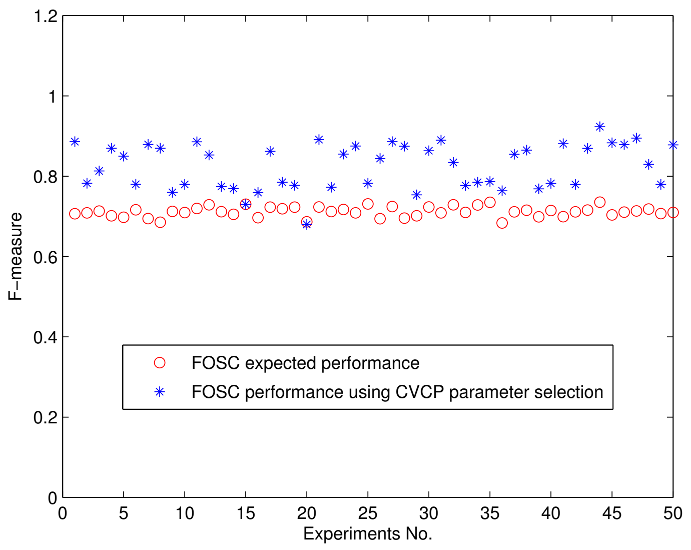


Figure 4.23: Iris data set, $\beta = 20\%$.

Table 4.19: Average scores for 50 experiments on Wine

Performance / β	CVCP-Q.	Expected-Q.	CVCP-std.	Expected-std.
5	0.4659	0.4580	0.0326	0.0049
10	0.4717	0.4569	0.0261	0.0161
20	0.5569	0.5127	0.0338	0.0191

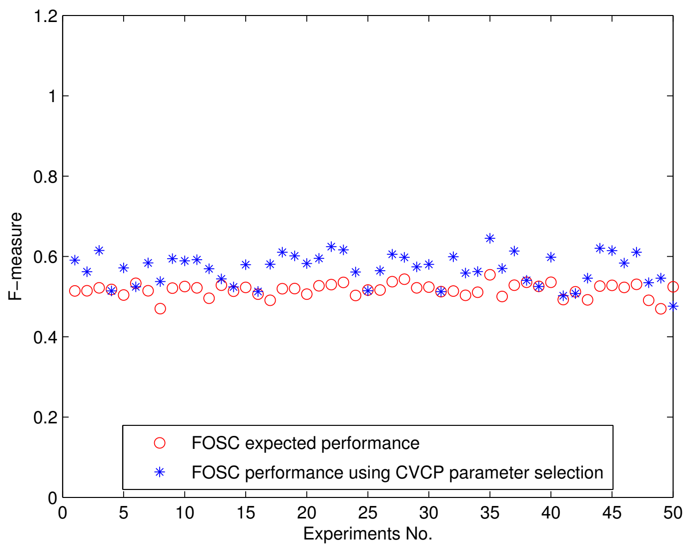


Figure 4.24: Wine data set, $\beta = 20\%$.

Table 4.20: Average scores for 50 experiments on Ionosphere

Performance	CVCP-Q.	Expected-Q.	CVCP-std.	Expected-std.
β				
5	0.6036	0.5328	0.0311	0.0063
10	0.6189	0.5738	0.0086	0.0065
20	0.6228	0.5181	0.0106	0.0088

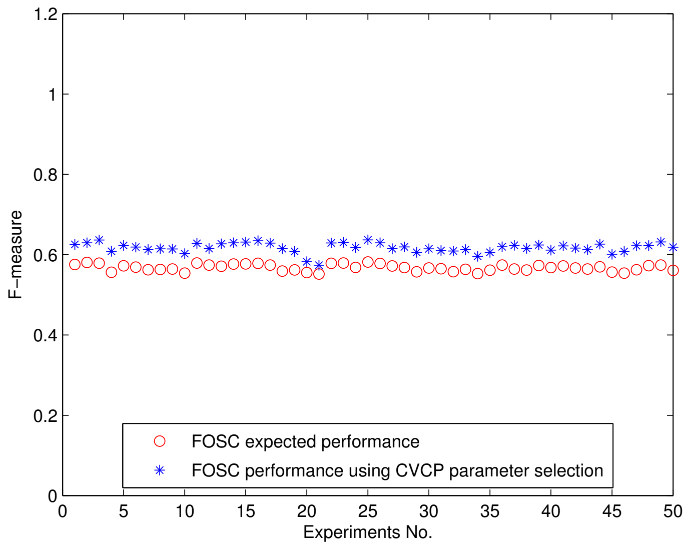


Figure 4.25: Ionosphere data set, $\beta = 20\%$.

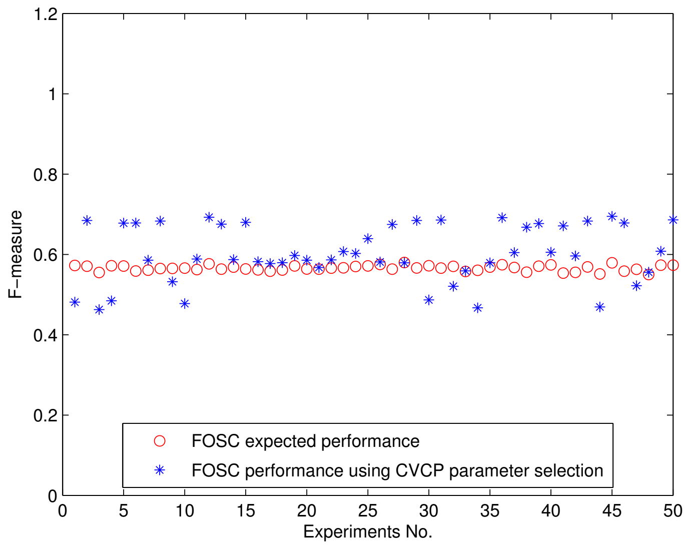


Figure 4.26: Ecoli data set, $\beta = 10\%$.

Table 4.21: Average scores for 50 experiments on Ecoli

Quality / Percent	CVCP-Q.	Expected-Q.	CVCP-std.	Expected-std.
5	0.6555	0.6532	0.0192	0.0040
10	0.6026	0.5659	0.0723	0.0071
20	0.5749	0.5668	0.0202	0.0112

The above results show improvements in the expected performances of FOSC when the appropriate value of MinPts is suggested by the CVCP. For Ionosphere data set, slight improvements happen when increasing the number of constraints which means that using the 5% of labeled points involved in constraints already provide a good partial supervision.

4.5.2 MPCKMeans performance using the CVCP selected k

This section explains how CVCP improves MPCKmeans performance by finding the "best" value of k. The procedure 6 shows the pseudo code of the CVCP evaluation using MPCKmeans. The MPCKmeans performance using the CVCP selected k is compared with the expected performance of MPCKmeans and the MPCKmeans performance using the parameter k selected based on the Silhouette width [10]. The Silhouette width is one of the well-balanced methods in the literature to estimate the value of k [16] [29] [30]. The parameter selected by Silhouette width is the parameter that has the maximum Silhouette width. We report the average results of this evaluation on different data sets over 50 experiments.

Procedure 6 Experiment: MPCKmeans performance using the CVCP selected parameter.

```
Input: Data set $X$, a fixed set of labeled objects $P$, range of parameter values
    $S$.
 1: Run CVCP for $X$ using constraints derived from $P$ to select the "best"
    value of $k$, $s'$.
 2: Run semi-supervised clustering algorithm with parameter $s'$: (MPCK-
    means performance using the CVCP selected parameter).
 3: for $s \in S$ do
 4:     Run MPCKmeans with $s$.
 5:     Record clustering quality for the output partition with respect to the
        ground truth (excluding the points in $P$).
 6:     Compute Silhouette width for the output partition.
 7:     Recod Silhouette width.
 8: end for
 9: Average all recorded clustering qualities (MPCKmeans expected perfor-
    mance).
10: Find the maximum recorded Silhouette width. (MPCKmeans performance
    using Silhouette coefficient).
```

Results

The average results of the MPCKmeans performances are reported on ALOI, Julia-Handl and UCI data sets. For data set collections, such as ALOI and Julia-Handl the average results of all data sets in the collections are reported. We also depict the results of procedure 6 over 50 experiments for one individual data set from these collections as representative examples. In tables and figures of this section, "CVCP-Q" and "CVCP-std" denote the average and standard deviation of the MPCKmeans performance using the CVCP selected k over 50 experiments. "Expected-Q" and "Expected-std" denote the average and standard deviation of the MPCKmeans expected performance over 50 experiments. "Sil-Q" and "Sil-std" denote the average and standard deviation of the MPCKmeans performance using Silhouette width over 50 experiments

ALOI

Figure 4.27 shows one average case representative example for the values of "MPCKmeans performance using the CVCP selected parameter" and "MPCKmeans expected performance" over 50 experiments for the data set $k5-11$ from the ALOI collections. Each experiment uses a different set of constraints.

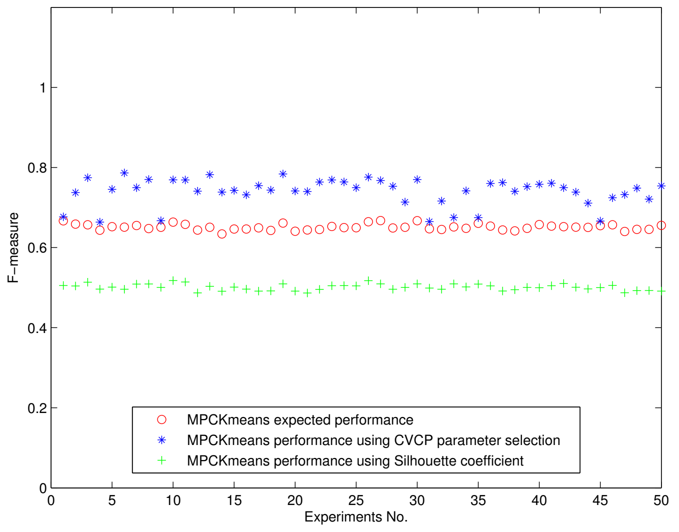


Figure 4.27: Data set $k5-11$ from ALOI, $\beta = 5\%$.

Table 4.22: Average scores for 50 experiments on ALOI

Performance β	CVCP-Q.	Expected-Q.	Sil-Q.	CVCP-std.	Expected-std.	Sil-std.
5	0.7001	0.6250	0.5875	0.0506	0.0045	0.0105
10	0.7196	0.6253	0.5876	0.0485	0.0077	0.0103
20	0.7290	0.6271	0.5881	0.0410	0.0131	0.0162

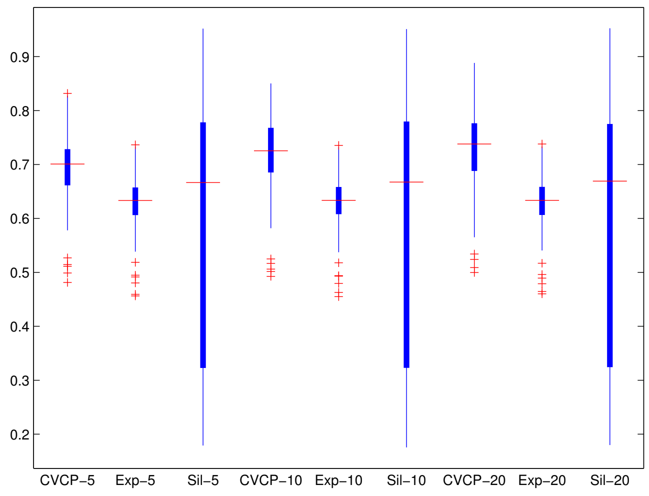


Figure 4.28: Boxplot of CVCP-Q, Expected-Q and Silhouette-Q with different number of constraints - ALOI collection.

Table 4.22 shows results for different number of constraints for the collection ALOI. The results reported in table 4.22 are averaged of 100 data sets in the ALOI collection. Figure 4.28 depicts the boxplot of values in table 4.22. We can see a significant improvement in MPCKmeans performance when using the "best" k based on the CVCP framework.

Julia-Handl

Figure 4.29 show one best case representative example for the values of "MPCKMeans performance using the CVCP selected parameter", "MPCKMeans expected performance" and "MPCKmeans performance using Silhouette coefficient" over 50 experiments for the data sets no.7 from the 2D Julia-Handl collection. Figure 4.31 show one average case representative example for the values of "MPCKMeans performance using the CVCP selected parameter", "MPCKMeans expected performance" and "MPCKmeans performance using Silhouette width" over 50 experiments for the data sets no.7 from the 10D Julia-Handl collection. In this case, the "MPCKMeans performance using the CVCP selected parameter" and "MPCKmeans performance using Silhouette width" are very close.

Tables 4.23 and 4.24 show the average results of 10 data sets from the 2 dimensional Julia-Handle collection and 0 data sets from the 10 dimensional Julia-Handle collection, respectively. Figures 4.30 and 4.32 depict the boxplots of values in table 4.23 and 4.24, respectively.

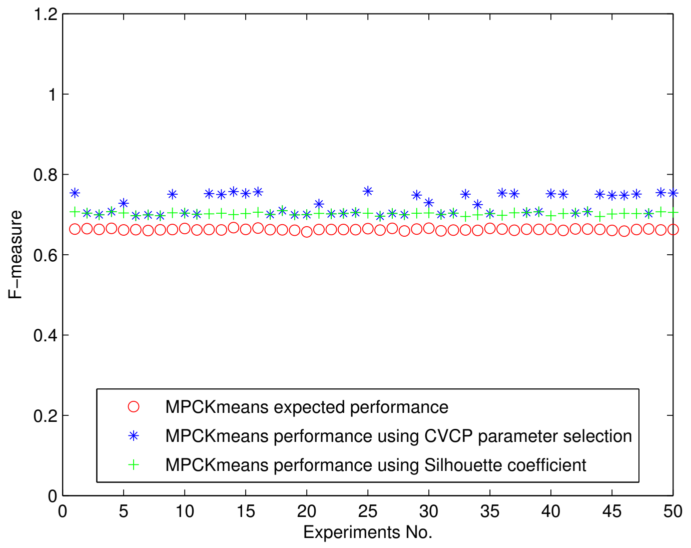


Figure 4.29: Data set no.7 in Julia-Handle 2D, $\beta = 5\%$.

Table 4.23: Average scores for 50 experiments on 2D Julia-Handl.

Performance / β	CVCP-Q.	Expected-Q.	Sil-Q.	CVCP-std.	Expected-std.	Sil-std.
2	0.7371	0.6811	0.6978	0.0494	0.0010	0.0014
5	0.7839	0.6810	0.6984	0.0376	0.0014	0.0025
10	0.7992	0.6810	0.6981	0.0224	0.0020	0.0035

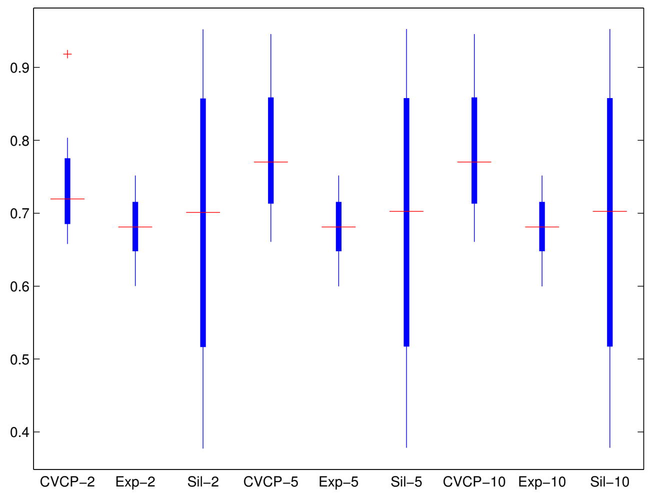


Figure 4.30: Boxplot of CVCP-Q, Expected-Q and Silhouette-Q with different number of constraints - 2D Julia-Handle collection.

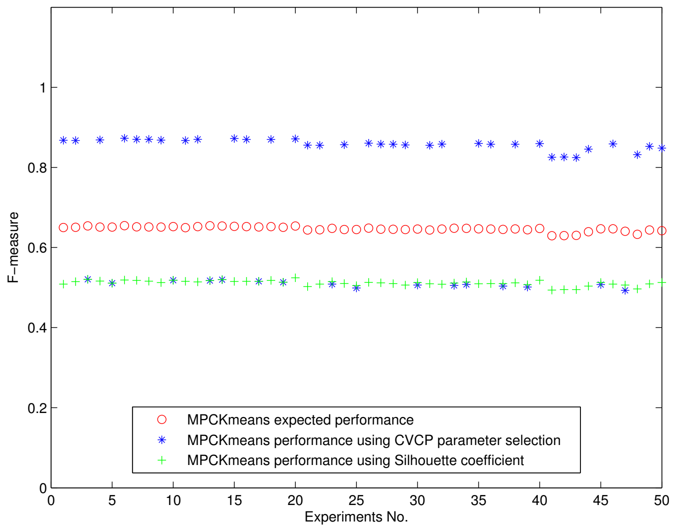


Figure 4.31: Data set no.8 from Julia-Handle 10D, $\beta = 5\%$.

Table 4.24: Average scores for 50 experiments on 10D Julia-Handl.

Performance / β	CVCP-Q.	Expected-Q.	Sil-Q.	CVCP-std.	Expected-std.	Sil-std.
2	0.7139	0.6514	0.6319	0.0765	0.0011	0.0018
5	0.7569	0.6517	0.6321	0.0577	0.0019	0.0028
10	0.8056	0.6499	0.3701	0.0186	0.0025	0.0077

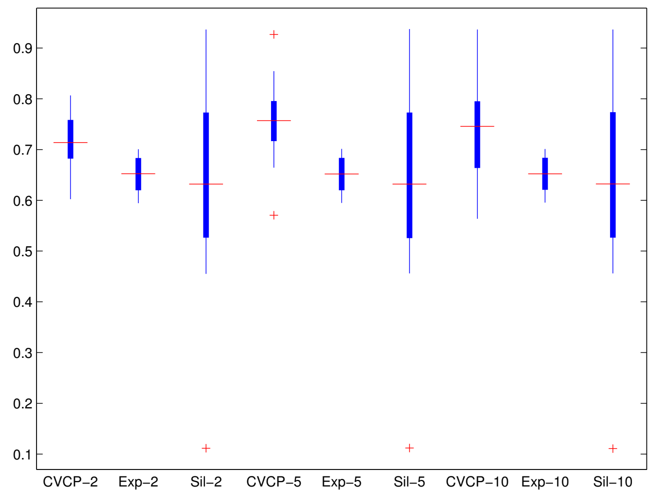


Figure 4.32: Boxplot of CVCP-Q, Expected-Q and Silhouette-Q with different number of constraints - 10D Julia-Handle collection.

We can see that the MPCKMeans performance on Julia-Handl collections are improved by using the CVCP suggested parameters. As it is shown in the boxplots, “CVCP-Q” outperforms “Expected-Q” and “Silhouette˙Q” for different number of constraints.

UCI

Figures 4.33, 4.34 , 4.35 and 4.36 show the values of “MPCKmeans using the CVCP selected parameter” and “MPCKmeans expected performance” and “MPCKmeans performance using Silhouette width” over 50 experiments for the data sets in UCI repository; Iris, Wine, Ionosphere and Ecoli respectively. Tables 4.25, 4.26, 4.27 and 4.28 show the results for different number of constraints for these data sets.

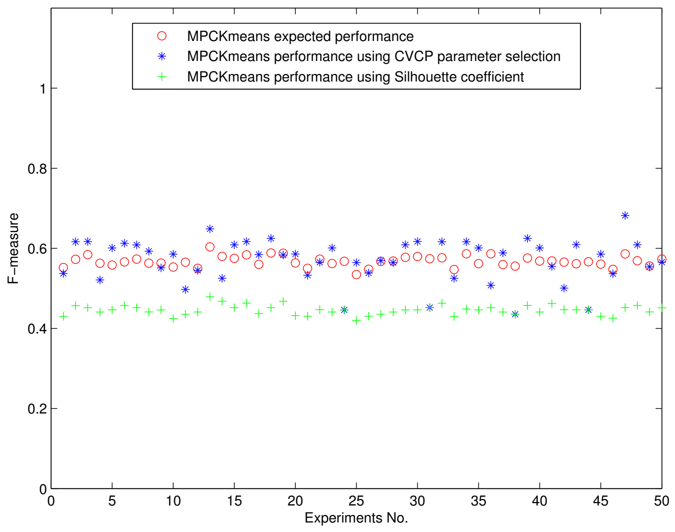


Figure 4.33: Iris data set, $\beta = 20$

Performance / β	CVCP-Q.	Expected-Q.	Sil-Q.	CVCP-std.	Expected-std.	Sil-std.
5	0.5585	0.5649	0.4456	0.0452	0.0055	0.0057
10	0.5475	0.5645	0.4444	0.0492	0.0083	0.0066
20	0.5697	0.5676	0.4457	0.0539	0.0132	0.0122

Table 4.25: Average scores for 50 experiments on Iris

Performance / β	CVCP-Q.	Expected-Q.	Sil-Q.	CVCP-std.	Expected-std.	Sil-std.
5	0.6523	0.6341	0.3772	0.0416	0.0045	0.0054
10	0.6392	0.6334	0.3753	0.0425	0.0073	0.0099
20	0.6397	0.6367	0.3777	0.0278	0.0106	0.0124

Table 4.26: Average scores for 50 experiments on Wine

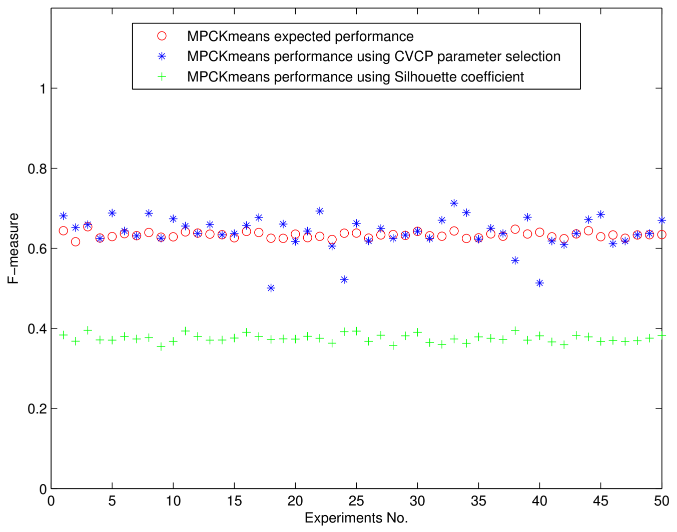


Figure 4.34: Wine data set, $\beta = 10$

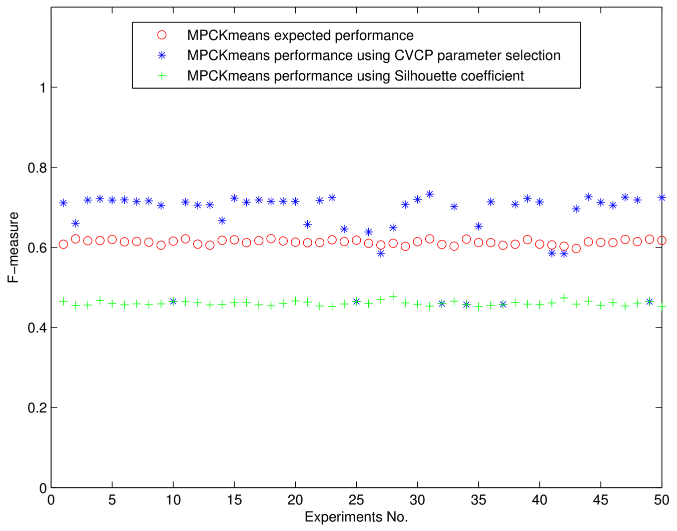


Figure 4.35: Ionosphere data set, $\beta = 10$

Performance	CVCP-Q.	Expected-Q.	Sil-Q.	CVCP-std.	Expected-std.	Sil-std.
β						
5	0.6159	0.6119	0.4602	0.0641	0.0051	0.0039
10	0.6681	0.6129	0.4601	0.0853	0.0059	0.0054
20	0.6857	0.6133	0.4605	0.0729	0.0078	0.0079

Table 4.27: Average scores for 50 experiments on Ionosphere

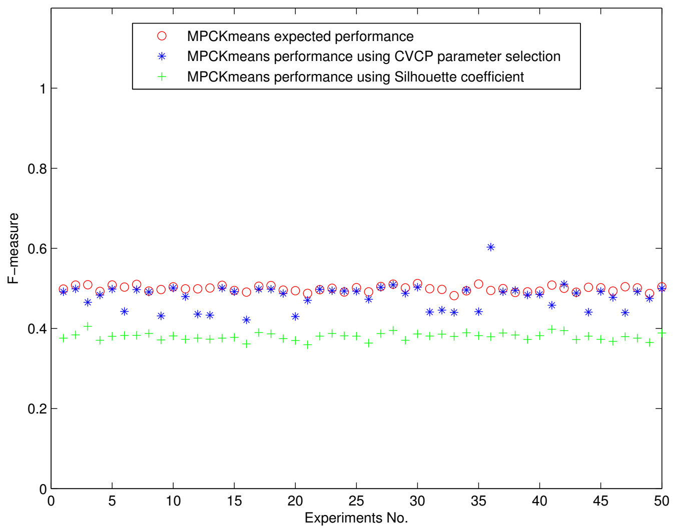


Figure 4.36: Ecoli data set, $\beta = 20$

Performance / β	CVCP-Q.	Expected-Q.	Sil-Q.	CVCP-std.	Expected-std.	Sil-std.
5	0.4914	0.5025	0.3783	0.0812	0.0037	0.0046
10	0.4705	0.5021	0.3785	0.0719	0.0051	0.0063
20	0.4800	0.4992	0.3798	0.0310	0.0072	0.0093

Table 4.28: Average scores for 50 experiments on Ecoli

CVCP improves the MPCKmeans performances on the UCI data sets except for Iris. As it was in the section 4.4.2, the MPCKmeans can not find clusters the ground truth classes accurately given the natural number of classes and it is not a suitable method to partition the Iris data. In this case, CVCP cannot improve the clustering performance.

4.6 Effects of increasing the number of constraints

We reported results for different number of constraints generated directly from some labeled data points. Some of the results in this chapter showed that adding more constraints improves the average clustering performances whereas other results showed that the average clustering performance have dropped. The inverse effects of more number of constraints had been studied by [31]. Their empirical results showed that, surprisingly, some constraint sets actually decrease clustering performance. This behavior is not caused by noise or errors in the constraints but its a results of the interaction between a given set of constraints and the clustering algorithm [31]. The paper introduces two measures, informativeness and coherence to explain the ill effect of some constraints.

Definition 9. Informativeness: refers to the amount of information in the constraint set that the algorithm cannot determine on its own. It is determined by the clustering algorithm's objective function and search preference [31].

If a clustering algorithm is biased towards grouping nearby points together and separating distant points, the the below constraints contradict this bias. The constraints in figure 4.37 is highly informative and can have a considerable impact on the way that the clustering algorithm partitions the points that are involved in these constraints.

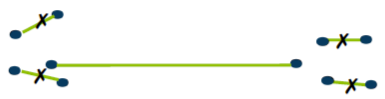


Figure 4.37: Sample illustrative example of constraints with high informativeness for the clustering algorithm CKM

The second measure, coherence, is defined as follows:

Definition 10. Coherence: measures the amount of agreement within the constraints themselves, with respect to a given distance metric [31].

Some constraints in the constraint set impose a contradictory metric for some close points. The sample constraints in figure 4.38 shows two parallel ML and CL constraints. The ML constraint indicates the distance between the points and surrounding points should be small. However, the CL constraint imposes that the distance between these points should be large. These two constraints represent an incoherent constraint set with respect to a Euclidean distance metric. Incoherent constraint sets may lead to select bad initial clusters and decreasing clustering performance.

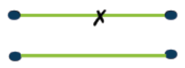


Figure 4.38: Sample illustrative example of constraints with low coherence, given a Euclidean distance metric.

Empirical results in [31] shows that the constraint sets with high informativeness and coherence are most likely to provide performance gains.

4.7 Conclusion from results

We presented results for two different type of evaluations in this chapter. The first evaluation method investigated the correlation of clustering scores of a semi-supervised clustering method with internal classification scores obtained within the CVCP. This evaluation showed strong correlations on all of the data sets for FOSC and MPCKmeans method. Generally, the evaluation showed weak correlations where the clustering method is not able to partition data well for the number of natural classes.

The second evaluations investigated the improvements in clustering performance when the parameter value is selected by CVCP. The FOSC performance based on the CVCP selected parameter outperformed the FOSC expected performance nearly in all the data sets. The MPCKmeans performance based on the CVCP selected parameter outperformed the MPCKmeans expected performance in most the data sets except for the Iris data set that obtained poor correlation values. The MPCKmeans performance based on the CVCP selected parameter outperformed the clustering performance based on Silhouette width for all the data sets. At extreme cases where the clustering method performs clustering very well or very poor regardless of the value of the parameter, the clustering performance obtained by CVCP is the about the same as the expected performance.

Chapter 5

Conclusions and Future Work

In this thesis, we addressed the problem of finding the "best" parameter value for different clustering methods (e.g., k for K-means based clustering methods, MinPts for density-based clustering methods). The choice of the parameter value can be very important for the clustering performance depending on an algorithm's sensitivity to the parameter. We discussed existing methods that estimate the "best" parameter in partitional and density-based clustering paradigms. Most of these approaches attempt to find the "best" parameter value for unsupervised clusterings using some internal validation measures.

Few researchers have studied the problem of an appropriate parameter selection for semi-supervised clustering methods. However, the fact that semi-supervised clustering algorithms take as input a small number of labeled objects or a small number of Must-link and Cannot-link constraints often provide additional opportunities for the parameter value selection. In our work, we introduce a general method, denoted as "Cross-validation framework for finding clustering parameters" (CVCP). Given a data set, CVCP selects the "best" parameter value for a semi-supervised clustering method based on available labels that are given as input to the semi-supervised clustering method. The CVCP framework evaluates a semi-supervised clustering performance for a parameter value by measuring the constraints satisfaction of the given constraints in the clustering results. We employed a cross-validation framework to obtain a robust estimate of a constraint satisfaction score for each value of a parameter. The parameter value that has the highest constraints satisfaction score is selected as the "best".

We evaluated the CVCP framework in two different ways and reported results for real and synthetic data sets such as ALOI, UCI and Julia-Handl. The evaluations were designed to identify the CVCP usefulness for finding the "best" value of MinPts and k for the clustering methods FOSC and MPCKMeans, respectively. As the first evaluation method, we investigated the correlation of the semi-supervised clustering performances for different values of the parameter with the CVCP's internal classification scores. The experiments showed that generally the clustering performances and the internal classification scores have strong correlations.

For the second evaluation method, We ran the FOSC method using the CVCP selected MinPts value and then evaluated how well the output clustering partition conformed to the ground truth. We compared the FOSC performance obtained using the CVCP selected MinPts value to the FOSC expected performance when the MinPts value has to be guessed. The experiments showed that the FOSC performance obtained using the CVCP selected MinPts value outperformed the expected performance.

In case of MPCKmeans, we ran MPCKmeans using the CVCP selected value of k and then evaluated how well the output clustering partition conformed to the ground truth. We compared the MPCKmeans performance obtained using the CVCP selected value of k to the MPCKmeans expected performance when the value of k has to be guessed. In addition, We compared the MPCKmeans performance obtained using the CVCP selected value of k to the MPCKmeans performance obtained using the parameter value k selected based on the Silhouette width. The experiments showed that the MPCKmeans performance obtained using the CVCP selected value of k outperformed the expected performance generally. In addition, the MPCKmeans performance obtained using the CVCP selected value of k significantly improved the MPCKmeans performance based on the Silhouette width.

The CVCP framework can improve the performance of a semi-supervised clustering method if the clustering method is capable of improving results by choosing an appropriate parameter.

The future research can be in few different directions; our proposed framework uses a subset of labeled objects as an input in order to avoid the "overlapping" constraints between the training set and test set in the cross-validation. It can be studied the possibility of using the user-provided constraints directly into our proposed framework by solving the problem of implicit use of training fold constraints in the test fold.

Another future research can be evaluating the performance of our proposed framework in other scenarios such as finding multiple parameters in clustering algorithms (e.g., ε and MinPts), finding minimum cluster ratio parameter value in Auto Cluster (Automatic cluster extraction by Sander et al. [32]) or finding the scaling parameter value σ in Spectral clustering.

Bibliography

[1] V. K. Pang-Ning Tan, Michael Steinbach, *Introduction to Data Mining*. Harlow, GB: Pearson Education Limited, 2005.

[2] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, p. 14, California, USA, 1967.

[3] A. K. Jain and R. C. Dubes, *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[4] Y. Zhao, G. Karypis, and U. Fayyad, "Hierarchical clustering algorithms for document datasets," *Data Mining and Knowledge Discovery*, vol. 10, no. 2, pp. 141–168, 2005.

[5] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *KDD*, 1996.

[6] N. Grira, M. Crucianu, and N. Boujemaa, "Unsupervised and semi-supervised clustering: a brief survey," *A review of machine learning techniques for processing multimedia content, Report of the MUSCLE European Network of Excellence (FP6)*, 2004.

[7] C. Yang, X. Zhang, L. Jiao, and G. Wang, "Self-tuning semi-supervised spectral clustering," in *International Conference on Computational Intelligence and Security, CIS.2008.141*, IEEE, 2008.

[8] M. M.-T. Chiang and B. Mirkin, "Intelligent choice of the number of clusters in k-means clustering: an experimental study with different cluster spreads," *Journal of classification*, vol. 27, no. 1, pp. 3–40, 2010.

[9] R. Tibshirani, G. Walther, and T. Hastie, "Estimating the number of clusters in a data set via the gap statistic," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 2, pp. 411–423, 2001.

[10] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[11] T. Caliński and J. Harabasz, "A dendrite method for cluster analysis," *Communications in Statistics-theory and Methods*, vol. 3, no. 1, pp. 1–27, 1974.

[12] E. H. Ruspini, "Numerical methods for fuzzy clustering," *Information Sciences*, vol. 2, no. 3, pp. 319–350, 1970.

[13] D. Pelleg, A. W. Moore, *et al.*, "X-means: Extending k-means with efficient estimation of the number of clusters.," in *Proc. 17th Int. Conf. Machine Learning (ICML00)*, pp. 727–734, 2000.

[14] K. S. Pollard and M. J. Van Der Laan, "A method to identify significant clusters in gene expression data," *U.C. Berkeley Division of Biostatistics Working Paper Series*, 2002.

[15] S. Dudoit and J. Fridlyand, "Bagging to improve the accuracy of a clustering procedure," *Bioinformatics*, vol. 19, no. 9, pp. 1090–1099, 2003.

[16] L. Kaufman and P. J. Rousseeuw, *Finding groups in data: an introduction to cluster analysis*, vol. 344. Wiley. com, 2009.

[17] L. Hubert and P. Arabie, "Comparing partitions," *Journal of classification*, vol. 2, no. 1, pp. 193–218, 1985.

[18] M. Bilenko, S. Basu, and R. J. Mooney, "Integrating constraints and metric learning in semi-supervised clustering," in *Proceedings of the twenty-first international conference on Machine learning*, pp. 11–, ACM, 2004.

[19] R. Campello, D. Moulavi, A. Zimek, and J. Sander, "A framework for semi-supervised and unsupervised optimal extraction of clusters from hierarchies," *Data Mining and Knowledge Discovery*, pp. 1–28, 2013.

[20] S. Basu, A. Banerjee, and R. J. Mooney, "Semi-supervised clustering by seeding," in *Proceedings of the twenty-first international conference on Machine learning*, vol. 2, pp. 27–34, 2002.

[21] C. Ruiz, M. Spiliopoulou, and E. Menasalvas, "C-dbscan: Density-based clustering with constraints," in *Rough Sets, Fuzzy Sets, Data Mining and Granular Computing*, pp. 216–223, Springer, 2007.

[22] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical association*, vol. 66, no. 336, pp. 846–850, 1971.

[23] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl, "Constrained k-means clustering with background knowledge," in *in Proc. 8th Int. Conf. Machine Learning*, pp. 577–584, 2001.

[24] A. Strehl, J. Ghosh, and R. Mooney, "Impact of similarity measures on web-page clustering," in *Workshop on Artificial Intelligence for Web Search (AAAI 2000)*, pp. 58–64, 2000.

[25] J. M. Geusebroek, G. J. Burghouts, and A. W. M. Smeulders, "The amsterdam library of object images," *International Journal of Computer Vision*, vol. 61, no. 1, pp. 103–112, 2005.

[26] D. Horta and R. J. Campello, "Automatic aspect discrimination in data clustering," *Pattern Recognition*, vol. 45, no. 12, pp. 4370–4388, 2012.

[27] C. Blake and C. J. Merz, "{UCI} repository of machine learning databases," 1998.

[28] J. Handl and J. Knowles, "Cluster generators for large high-dimensional data sets with large numbers of clusters," *Dimension*, vol. 2, p. 20, 2005.

[29] S. Dudoit and J. Fridlyand, "A prediction-based resampling method for estimating the number of clusters in a dataset," *Genome biology*, vol. 3, no. 7, p. research0036, 2002.

[30] Y. Liu, Z. Li, H. Xiong, X. Gao, and J. Wu, "Understanding of internal clustering validation measures," in *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pp. 911–916, IEEE, 2010.

[31] I. Davidson, K. L. Wagstaff, and S. Basu, *Measuring constraint-set utility for partitional clustering algorithms*. Springer, 2006.

[32] J. Sander, X. Qin, Z. Lu, N. Niu, and A. Kovarsky, "Automatic extraction of clusters from hierarchical clustering representations," in *Advances in Knowledge Discovery and Data Mining*, pp. 75–87, Springer, 2003.