GRID-BASED SUPERVISED CLUSTERING ALGORITHM USING GREEDY AND GRADIENT DESCENT METHODS TO BUILD CLUSTERS

Pornpimol Bungkomkhun

A Dissertation Submitted in Partial
Fulfillment of the Requirements for the Degree of
Doctor of Philosophy (Computer Science)
School of Applied Statistics
National Institute of Development Administration
2012

GRID-BASED SUPERVISED CLUSTERING ALGORITHM USING GREEDY AND GRADIENT DESCENT METHODS TO BUILD CLUSTERS

Pornpimol Bungkomkhun

School of Applied Statistics

Associate Professor ..... Advisor
(Surapong Auwatanamongkol, Ph.D.)

The Examining Committee Approved This Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy (Computer Science).

Associate Professor .....Committee Chairperson
(Pipat Hiranvanichakorn, Ph.D.)

Associate Professor ..... Committee
(Surapong Auwatanamongkol, Ph.D.)

Assistant Professor ..... Committee
(Ohm Sornil, Ph.D.)

Assistant Professor .....Committee
(Rawiwan Tenissara, Ph.D.)

Instructor ..... Dean
(Lersan Bosuwan, Ph.D.)

31 July 2012

ABSTRACT

Title of Dissertation	Grid-based Supervised Clustering Algorithm Using Greedy and Gradient Descent Methods to Build Clusters
Author	Mrs. Pornpimol Bungkomkhun
Degree	Doctor of Philosophy (Computer Science)
Year	2012

Clustering analysis is one of the primary methods of data mining tasks with the objective to understand the natural grouping (or structure) of data objects in a dataset. The clustering tasks aim to segment the entire data set into relatively homogenous subgroups or clusters where the similarities of the data objects within clusters are maximized and the similarities of data objects belonging to different clusters are minimized. For supervised clustering, not only attribute variables of data objects but also the class variable of data objects take part in grouping or dividing data objects into clusters in the manner that each cluster has high homogeneity in term of classes of its data objects.

This dissertation proposes a grid-based supervised clustering algorithm that is able to identify clusters of any shapes and sizes without presuming any canonical form for data distribution. The algorithm not only needs no pre-specified number of clusters but also is insensitive to the order of the input data objects. The proposed algorithm gradually partitions data space into equal-size grid cells using one dimension at a time. The greedy method is used to arrange the order of dimensions for the gradual partitioning that would give the best quality of clustering, while the gradient descent method is used to find the optimal number of intervals for each partitioning. After all dimensions have been partitioned, any connected dense grid cells containing majority of data objects from the same class are merged into a cluster. By using the greedy and gradient descent methods as mentioned, the proposed algorithm can produce high quality clusters while reduce time to find the best

partitioning and avoid the memory confinement problem during the process. On two-dimensional synthetic datasets, the proposed algorithm can identify clusters with different shapes and sizes correctly. The proposed algorithm also outperforms other five supervised clustering algorithms when performed on some UCI datasets.

ACKNOWLEDGEMENTS

The author would like to express supreme thank to my advisor, Associate Professor Dr. Surapong Auwatanamongkol, for his valuable advice and guidance in every time of my irresolute conditions, and also for his intense encouragement whenever I got enervated.

The author also wish to extend thanks and appreciation to all of the committee members, Associate Professor Dr. Pipat Hiranvanichakorn, Assistant Professor Dr. Ohm Sornil and Assistant Professor Dr. Rawiwan Tenissara, for their beneficial comments and suggestions all through the progress of the dissertation.

Thank is also dedicated to Assistant Professor Dr. Apirak Jirayusakul for sending me the synthetic datasets to be used in the experiments without any hesitation.

Mrs. Pornpimol Bungkomkhun

July 2012

TABLE OF CONTENTS

LIST OF TABLES

Tables **Page**

LIST OF FIGURES

Figures **Page**

CHAPTER 1

INTRODUCTION

1.1 Problem Overview and Motivation

Rapid advances in data collection methodologies have enabled the accumulation of vast amount of data. Extracting meaningful information from these data has been very beneficial as well as challenging. Data mining is the process of automatically discovering useful information in large data repositories. Clustering analysis is one of the primary methods of data mining tasks with the objective to understand the natural grouping (or structure) of data objects in a dataset. The main objective of clustering is to separate data objects into high quality groups (or clusters), based on similarities among the data objects. The clustering tasks aim to segment the entire data set into relatively homogeneous subgroups or clusters where similarities of data objects within any clusters are maximized and similarities of data objects that come from different clusters are minimized.

Cluster analysis has long played an important role in a variety of fields. In biology, cluster analysis has been spent in creating the taxonomy or other words the hierarchical classification of living things. More recently, biologists have applied clustering to analyze the large amount of genetic information, as for example finding groups of genes that have similar functions. Clustering methodologies have been used in various schemes in information retrieval. One of the examples is grouping the search results of the World Wide Web queries into a small number of clusters, each of which captures a particular aspect of the query, in order to assist the web explorers in gaining their relevant information.

Clustering has also been used in climate analysis in finding patterns in the atmosphere and ocean that have a significant impact on land climate. In psychology and medicine, cluster analysis has been used to identify the variations of illnesses in forms of their different sub-categories. Even in the business areas, clustering has been

used to analyze large amounts of information on current and potential customers so as to segment them into groups for approaching appropriate marketing activities.

Figure 1.1 illustrates the difficulty of deciding what constitutes a cluster (Tan, Steinbach and Kumar, 2006: 490-491). Twenty points in the figure represent twenty data objects that are divided into clusters in three different ways: two clusters in figure 1.1(b), four clusters in figure 1.1(c), and six clusters in figure 1.1(d). The boundaries of clusters are identified by the surrounding circles. Clusters depicted in figure 1.1(b) and 1.1(d) can be recognized apparently by human eyes, while clusters depicted in figure 1.1(c) may be not.

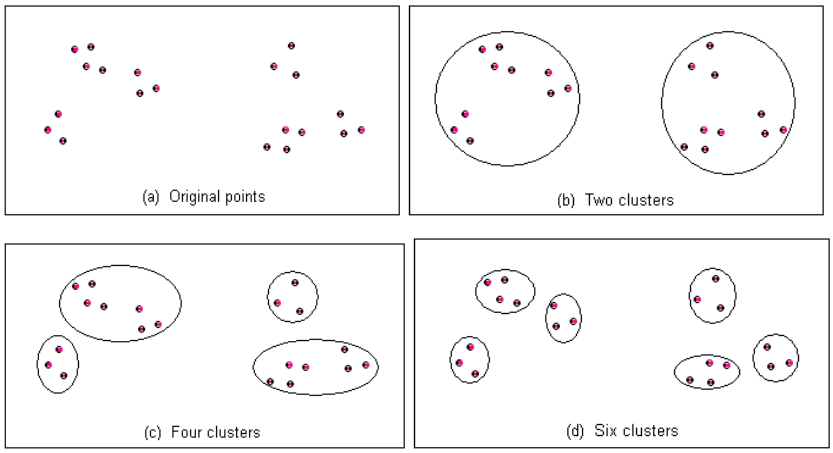


Figure 1.1 Different Ways of Clustering
Source: Adapted from Tan et al., 2005: 491.

Clustering techniques can be subdivided into three groups (Zeidat, Eick and Zhao, 2006: 1-2): traditional clustering, semi-supervised clustering, and supervised clustering. Traditional clustering is performed in an unsupervised learning manner. No class label attribute of data objects is used to guide clustering them into groups. Figure 1.2 gives the view of performing traditional clustering upon the twenty-eight demonstrative data objects. Four clearly tight clusters: A, B, C, and D are depicted. Since the problem of finding the optimal clustering of data objects was proven to be NP-complete (Fowler, Paterson and Tanimoto, 1981: 133-137; Megiddo and Supowit, 1984: 182-196 quoted in Procopiuc, 1997: 3), many heuristic methods have been

developed to solve the problem. Kotsiantis and Pintelas (2004: 74-78) categorized traditional clustering algorithms into several methods, namely, partitioning methods, hierarchical methods, density-based methods, grid-based methods, and model-based methods.

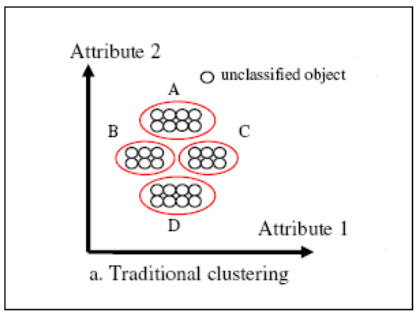


Figure 1.2 Example Result of Traditional Clustering Process
Source: Adapted from Zeidat et al., 2006: 2.

Apart from using only the similarity information, there are cases when a small amount of prior knowledge is available and is needed to be used as the guidance for the clustering process. Such knowledge concerns either in the form of the class labels of some data items or in the pair-wise constraints (in terms of ‘must-link’ or ‘cannot-link’) between some data items. In this situation, the semi-supervised clustering can be approached in order to focus not only on obtaining tight clusters but also on satisfying the given constraints with respect to a small set of data objects. Figure 1.3 exhibits the result of semi-supervised clustering on an example scenario. The clusters E, F, G, and H are generated where objects belonging to different classes are separated into different clusters, whereas those belonging to the same classes are grouped into the same clusters. Notice information of class labels of some objects are used to guide the semi-supervised clustering.

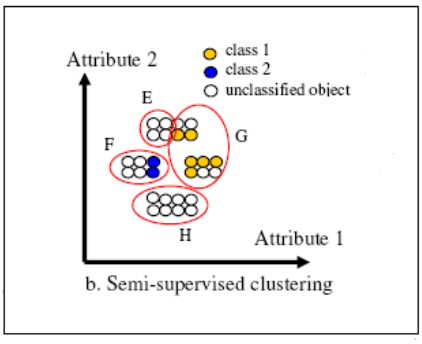


Figure 1.3 Example Result of Semi-supervised Clustering Process
Source: Adapted from Zeidat et al., 2006: 2.

Unlike the goal of traditional unsupervised clustering, the goal of supervised clustering is to identify class-uniform clusters that have high data densities (Zeidat et al., 2006: 3). According to them, not only data attribute variables, but also a class variable, take part in grouping or dividing data objects into clusters in the manner that the class variable is used to supervise the clustering. Supervised clustering differs from semi-supervised clustering in that the former method utilizes prior knowledge (in terms of class labels) of all data objects, while the latter method needs the prior knowledge of some data objects in a given dataset. At the end of supervised clustering process, each cluster is assigned with specific class label corresponding to the majority class of data objects inside the cluster. Eick, Zeidat and Zhenghong (2004: 774-776) proposed supervised clustering algorithm that uses a fitness function which maximizes the purity of clusters while keeping the number of clusters low. The example scenario describing the result of their supervised clustering is shown in figure 1.4 where data objects are grouped into four clusters: I, J, K, and L, each of which governs objects of the same class only.

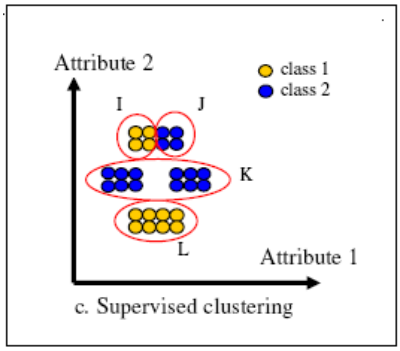


Figure 1.4 Example Result of Supervised Clustering Process
Source: Adapted from Zeidat et al., 2006**:** 2.

Supervised clustering concept has been used in various applications. Dettling (2003: 1-8) used supervised clustering technique in revealing groups of co-regulated predictor variables (genes) in microarray data which are controlled by external (supervised) information in order to predict a certain type of disease. His approach yielded more effective result than any unsupervised techniques clustering. Aggarwal, Gates and Yu (1999: 352-356) proposed methods for building categorization systems to categorize documents by applying supervised clustering, using a priori knowledge of the definition of each category. They built a categorization system using a set of classes from the Yahoo! Taxonomy and showed that the supervised clustering created a new set of classes which were surveyed to be as good as the original set of classes in the Yahoo! Taxonomy, but which are naturally suited to automated categorization. The result is a system which has much higher overall quality of categorization than systems based on manual taxonomies. Zeidat et al. (2006) and Eick et al. (2004) used supervised clustering information in creating background knowledge for a dataset. This knowledge was used to analyze the distribution of the instances of classes in the attribute space with the objective to discover the subclasses of each particular class. They also proposed supervised clustering as a tool to select the minimal subset of examples from a training dataset that would enhance the accuracy of a classifier. As simple classifiers are known to exhibit low variances and high biases, Zeidat et al. (2006) proposed using supervised clustering to enhance simple classifiers via class decomposition.

1.2 Dissertation Objectives

Until recently, there have been quite a few previous works in the area of supervised clustering algorithms. Due to the acknowledgment that no single clustering method can adequately handle all sorts of cluster structures and that different clustering approaches often define different definitions for clusters, it is impossible to define either a universal algorithms or a universal fitness function to measure clustering quality. However, most of the supervised clustering algorithms still contain one or the other drawbacks. For a number of algorithms, the clustering results may depend on the ordering of data objects that are presented, i.e. their result clusters may be not identical if the ordering of input data differs. Some algorithms need to have pre-specified number of clusters, or moreover the initialization of the cluster representatives, as the guidance in the clustering process. Other algorithms may decrease their clustering proficiency by restricting the result clusters to be within the global shape boundaries.

As mentioned in Kotsiantis et al. (2004: 79) that “A solution for better results could be instead of integrating all requirements into a single algorithm, to try to build a combination of clustering algorithms.”, the contribution of this dissertation is to propose the new idea of supervised clustering algorithm based on the combination of two methods, i.e. grid-based approach to density clustering method and bottom-up subspace clustering method. The proposed algorithm is intended to eliminate all mention drawbacks occurred in the previous algorithms. The algorithm gradually partitions data space into equal-size nonempty grid cells (containing data objects) using one dimension at a time for partitioning, and merges the connected grid cells with same data class majorities to form partial clusters until all dimensions have been partitioned. This process follows the framework of bottom-up subspace clustering. To gradually partition data space into nonempty grid cells, the proposed algorithm first finds the order of dimensions to be used for the gradual subspace clustering by considering each of the data dimensions as the only dimension for partitioning (individual dimension subspace clustering). The optimal number of equal intervals to achieve the best quality of supervised clustering for each partitioning of a dimension is determined using the gradient descent search instead of sequential search as

presented in Pornpimol Bungkomkhun and Surapong Auwatanamongkol (2009: 539-541). Next, the bottom up subspace clustering is performed by gradually partitioning data space using one dimension at a time. Using the greedy approach, dimensions are selected to be the next dimension for partitioning based on their clustering quality fitness values achieved from individual dimension clustering. The optimal individual dimension clustering is also performed using the same gradient descent method. Once all dimensions have participated in the partitioning, grid cells are formed and all connected nonempty cells containing the majority of data from a same class are merged into a cluster. Finally, each cluster is then labeled with its majority class of data.

CHAPTER 2

LITERATURE REVIEW

The proposed algorithm is the supervised clustering algorithm that relies on grid-based clustering method in quantizing data space into grid cells in the bottom-up fashion by gradually adding dimensions into the cells one at a time. All adjacent cells whose classes are identical are finally merged into the same cluster by applying density-based clustering concept. In this section, essential backgrounds on traditional clustering, subspace clustering and supervised clustering are provided. Reviews on clustering algorithms relevant to the proposed algorithm are also given.

2.1 Traditional Clustering

2.1.1 Partitioning Algorithm

The basic idea of partitioning algorithms is to partition the data objects into a set of *k* clusters. Usually, the algorithms start with an initial partition and afterwards use the iterative control strategy to assign each object to the closest cluster with regard to optimizing the defined objective function. There are mainly two approaches, *k-means* and *k-medoid* algorithms (Tan et al., 2006: 496-514), which are different in the way the clusters are represented. In the *k-means* algorithm, each cluster is represented by the center of its gravity, whereas in the *k-mediod* algorithm, one of the object located near the center of the cluster is used as the representative of the cluster. The partitioning algorithms are effective only in case of clusters that have convex shapes, similar sizes, and commensurate densities, and are under the reasonably estimated number of clusters (Kotsiantis et al., 2004: 79), however.

2.1.2 Hierarchical Algorithm

The hierarchical algorithms (Tan et al., 2006: 515-526) decompose data objects into hierarchical levels of partitioning, usually represented by a dendrogram -

a tree keeping track of the splitting data objects in each level. The algorithm recursively splits data objects into smaller subsets until the termination condition is satisfied. The dendrogram can be created in either the top-down manner (divisive approach) or in the bottom-up manner (agglomerative approach). Although having ability to handle clusters of different sizes, for high-dimensional data the hierarchical algorithms are expensive in terms of the computational and storage requirements (Kotsiantis et al., 2004: 79).

2.1.3 Density-based Algorithm

In density-based clustering (Tan et al., 2006: 526-532), clusters are defined as regions of higher density of data objects than their surrounding regions. The algorithms locate regions of high density that are separated from one another by regions of low density, and hence have major ability to form clusters of arbitrary shapes and sizes. The most popular density-based clustering is *DBSCAN* which proposes the idea that the density of the neighborhood of a given radius of each point in the same cluster has to exceed the specified threshold. The density-based algorithms, hence, can cope with clusters of any shapes and sizes, but still have trouble with high-dimensional data, especially when dealing with large datasets (Kotsiantis et al., 2004: 79).

2.1.4 Grid-based Algorithm

The idea of the grid-based clustering is to quantize data space into a number of multi-dimensional (hyper-rectangular) cells and then perform the desired operations on the data objects in each cell's boundary, one cell at a time. The grid-based clustering approaches are frequently used as the preliminary passageway for the density-based clustering in the manner that data are separated into groups in accordance with the cells' structures, and only data in the dense cells are processed, cell by cell. The clusters are finally formed from merging cells that are sufficiently dense. The benefit of the grid-based approach to density clustering algorithms is their fast processing time which depends on the number of grid cells only, not on the number of data objects (Sheikholeslami, Chatterjee and Zhang, 1998: 430).

Wang, Yang and Muntz (1997) proposed the grid-based multi-resolution clustering algorithm named STING (A Statistical Information Grid Approach to Spatial Data). The algorithm aims to facilitate the region-oriented query processing in such a manner that the classes of queries and clustering problems can be answered with no need to access the full database. The algorithm divides data space into multi-level rectangular cells represented by a hierarchical structure. Each cell in the parent level is recursively partitioned into four child cells at the next lower level until the size of the leaf cells is less than the pre-defined density of the cells. The statistical information regarding the objects in each grid cell is captured and stored within the cells' structures, hence the statistical parameters of the higher level cells can easily be computed from those of the lower level cells. When comes a spatial data mining query, the algorithm first determines a layer from which the query processing procedure is to start (it is not necessary to start with the root level), and then uses a top-down approach to answer the query. For each cell in the layer, the algorithm calculates the relevancy of the cell to the query at the specified confidence level, and the irrelevant cells are then removed. After finish examining the current layer, the cells in the next lower level of the relevant cells are repeated processed until the lowest layer is reached. Through this approach, instead of going through all cells, the algorithm looks only at those cells that are children of the relevant cells of the previous layer. The algorithm finally identifies all the regions formed by relevant cells and returns them as the answer to the query.

The WaveCluster (Sheikholeslami et al., 1998: 428-439) views the multi-dimensional data objects as a multi-dimensional signal. The algorithm begins by quantizing data space into multi-dimensional grid cells. The signal processing technique, the wavelet transformation, is then applied to convert the data objects in the region of each cell into the frequency domain. The WaveCluster identifies clusters by means of detecting the connected components in the transformed space, and afterwards map the cells in the transformed space back to the cells in the original space.

Hinneburg and Keim (1998) proposed the algorithm to cluster large multimedia databases called DENCLUE (DENsity-based CLUstEring). The influence function, a function describing the impact of a data point to its neighborhoods, is

applied to each data point. The overall density function is the sum of the influence function of all data points in the data space. Clusters are determined by identifying density-attractors, the local maxima of the overall density function, by means of a hill-climbing procedure guided by the gradient of the overall density function. The DENCLUE algorithm consists of two steps. The pre-clustering step is aimed to speed up the density function calculation by constructing a map of the relevant portion of the data space. The data space is divided into equal size *d*-dimensional hypercubes, with an edge-length of 2σ, and only the hypercubes containing data points are connected to their neighboring cubes. In the actual clustering step, the density-attractors and the corresponding density-attracted points are identified. Only the highly populated cubes and their neighboring cubes are considered in determining clusters.

Kunttu, Lepisto, Rauhamaa and Visa (n.d.) presents the hierarchical clustering algorithm using multi-resolution grid-based clustering approach to enable the image browsing and retrieval in hierarchical manner. The goal of image browsing is to show the user a view of representative images of the database content. The images can be browsed in different scales through moving back and forth between general view and more specific image groups in the database, represented by different levels in hierarchical structure. The algorithm enables users to either zoom in to select a specific image type of interest, or zoom out to browse the variations in the image contents. When a desired image type is found at a certain level, users can zoom in to closer scrutinize the relevant clusters. The cells at a selected level are merged into clusters of similar images using density-based clustering approach.

Liao, Liu and Choudhary (2004) proposed the idea that employing on a single-level uniform grid only may not sufficient to obtain rational clustering quality. They presented a multi-scale grid-based clustering algorithm using adaptive mesh refinement (AMR) technique to automatically create different resolution grids based on the regional density. According to the algorithm, the additional meshes with higher resolution are recursively defined on each high density region until the desired density is met. The AMR tree is created to represent the various levels of grid cells' densities in the manner that the denser regions are to be indicated by the nodes nearer to the leaves. The clusters are created by using regions indicated by the leaves of the tree as

the cluster pivots in inducing the data objects underneath the region of their parent nodes, on the minimum distance approach, to be the members of the clusters. In this way, the AMR clustering algorithm can detect nested clusters at different level of resolution.

Park and Lee (2004: 32-37) proposed a grid-based algorithm for clustering data elements generated continuously and rapidly as a data stream. The algorithm initially partitions data space into a set of mutually exclusive equal-size cells. Not the physical data elements, but only the distribution statistics of which, are kept in the corresponding grid cells. The statistics of each cell are updated whenever the new generated data elements are within its boundary. The cells are recursively split out when their support parameters are higher than a pre-defined value. One of the dimensions of the data space is chosen as a dividing dimension based on the statistics of the data elements in the cell, resulting in splitting the cell into mutually exclusive smaller cells. The range of a dense cell can be partitioned by one of the three methods: *μ-partition*, *σ-partition* and *hybrid-partition*. The clusters are finally identified as groups of adjacent dense cells.

2.2 Subspace Clustering

Data objects may be related in different ways when different subsets of dimensions are considered. Thus, different clusters might exist when different sets of dimensions of the data objects are used for clustering. Subspace clustering aims to reveal clusters lying in various subspaces of the dataset. Parsans, Haque and Liu (2004: 90-105) classified subspace clustering algorithms into two major groups with regard to the search technique employed: the bottom-up search method and the top-down search method.

The bottom-up method searches for clusters from subspaces with smaller subsets of dimensions to subspaces with larger subsets of dimensions, whereas the top-down method does the other way around. Most of the bottom-up method used grid-based clustering in finding dense units in all different lower dimensional subspaces. In subsequently locating dense units in any higher dimensional subspaces, the downward closure property of density was applied, in an APRIORI style, in order

to reduce the search space. Clusters were lastly formed by combining all adjacent dense units together. Most of the researches in this field assume the search space to be restricted to axis-parallel subspaces.

A number of subspace clustering algorithms were categorized and reviewed in Parsans et al. (2004). One of them that the proposed algorithm is based on is CLIQUE (Agrawal, Gehrke, Gunopulos and Raghavan, 1998). CLIQUE is one of the very first subspace clustering algorithms. It is a grid-based clustering algorithm that provides an efficient approach for bottom-up subspace clustering. It uses an APRIORI style technique to find clusters in subspaces, based on the observation that dense areas in a higher-dimensional space imply the existence of dense areas in lower-dimensional space.

CLIQUE identifies dense clusters in a subspace of maximum dimensionality by automatically identifying projections of the data objects onto various subsets of dimensions where regions of high density with respect to those objects reside. The algorithm uses bottom-up approach in generating grid cells and identifying dense cells. It begins by finding dense units in all one-dimensional spaces, corresponding to each individual attribute of dataset. The algorithm then proceeds level-by-level, in the manner that the candidate *k*-dimensional dense cells can be determined using already determined *(k-1)*-dimensional dense cells. Hence the set of candidate *k*-dimensional cells that might possibly be dense can be found inside dense *(k-1)*-dimensional cells only. The algorithm is terminated when no more candidates are discovered. To form clusters, CLIQUE uses a depth-first search algorithm to find the connected dense cells, then creates cluster descriptions in the form of DNF expression.

2.3 Supervised Clustering

Supervised clustering is applied on classified data objects with the aim of identifying identical-class clusters that have high densities and have minimal impurity, with respect to majority classes of the clusters. The clustering is performed on attribute variables under the supervision of a target class variable. As a consequence, each generated cluster is labeled with only one specific class that has

majority data objects inside the cluster. Supervised clustering procedure is therefore used not only for knowledge discovery, but also for data classification, as the cluster structure with class information can be used as a classification function.

Tishby, Pereira and Bialek (1999: 368-377), Slonim and Tishby (1999: 617-623), and Aguilar, Ruiz, Riquelme and Giráldez (2001: 207-216) proposed supervised clustering algorithms based on bottom-up agglomerative approach. The algorithm proposed in Sinkkonen, Kaski and Nikkila (2002: 418-430) intended to find clusters that are homogenous in target class variable using a probabilistic approach based on discriminative clustering to minimize distortion within clusters. Qu and Xu (2004: 1905-1913) introduced supervised model-based clustering algorithms that were based on multivariate Gaussian mixture model which employs EM algorithm to estimate model parameters.

Finley and Joachims (2005: 217-224) proposed that supervised clustering can be achieved by training a clustering algorithm to produce desirable clusters. An SVM algorithm that could learn from an item-pair similarity measure to optimize clustering performance based on a variety of performance measures was proposed. Al-Harbi and Rayward-Smith (2006: 219-226) introduced supervised K-mean algorithm that combined Simulated Annealing with K-mean algorithm.

CCAS algorithms were developed for detecting intrusions into computer network system through intrusion signature recognition. The algorithms start by learning data patterns based on supervised clustering procedure, and afterwards uses these patterns for data classification. The original version of CCAS, namely CCA-S (Clustering and Classification Algorithm - Supervised) (Ye and Li, 2001: 1-4), starts with two dummy clusters and allows clusters of each individual class to spread over the entire data space regardless of the training sequence of data objects. Li and Ye (2002: 231-242) modified original CCAS with grid-based method to limit the search space in splitting training data objects into smaller size clusters. The algorithm begins with dividing data space into equal size grid cells. It then performed dummy-based approach only on data objects lying in the same cell.

Li and Ye (2005: 498-509) enhanced the robustness of CCAS by strengthening the algorithm with three post-processing steps: data redistribution, supervised grouping of clusters, and removal of outliers. ECCAS (Li and Ye, 2006: 396-406)

enabled CCAS to handle data of the mixed type, by introducing two methods for combining numerical and nominal variables in calculating distance measure. The first method combines different distance measure for each type of variables into a single distance measure ranging between 0 and 1. The second method is based on conversion of nominal variables to binary variables, and then treats these binary variables as numeric variables.

Three representative-based supervised clustering algorithms were introduced in Eick et al. (2004: 774-776) and Zeidat et al. (2006): Supervised Partitioning Arround Medoids (SPAM), Single Representative Insertion/Deletion Steepest Decent Hill Climbing with Randomized Start (SRIDHCR), and Supervised Clustering using Evolutionary Computing (SCEC). In their paper, the new fitness function used for measuring the algorithms was proposed. Instead of relying only on the tightness of objects in each cluster, like most of the traditional clustering algorithms, the three algorithms weights cluster purity against the number of generated clusters in the proposed fitness function.

SPAM, aimed to be the variation of PAM (Partitioning Around Medoids) that uses the proposed fitness function, starts by randomly selecting a mediod of the most frequent class data objects as the first representative. The algorithm then fills up the initial set of representatives with non-representative objects. The number of representatives is fixed by a pre-defined figure. SPAM later on repeatedly explores all possible replacements of a single representative of the most current solution by a single non-representative, provided that the new set of representatives induces minimum fitness function value. The algorithm terminates if none of the replacement can provide lower fitness function value.

In order to eliminate the limitation of SPAM that the number of representatives must be fixed by the pre-defined parameter, SRIDHCR algorithm permits either adding or removing any representatives into or from the current set of cluster representatives, with respect to the proposed fitness function. SRIDHCR is designed to be run r pre-specified times, with the intention to report the set of representatives that produces the lowest value of fitness function as the final result. At the beginning of each run, the algorithm randomly selects a pre-defined number of objects into the set of representatives. It afterwards repeatedly computes the fitness

function of the intermediate set of clusters, resulting from either adding a single non-representative into or removing a single representative from the current set of representatives. Each repeated loop ends by replacing the current representative set by the new set of representatives whose fitness function value is the lowest, providing that the value is less than those of the current representative set. In each run, the algorithm terminates whenever there is no significant improvement in the solution quality (measured by the value of the fitness function).

Besides the above two greedy algorithms, Zeidat et al. (2006) also proposed an evolutionary computing algorithm called SCEC. The algorithm evolves a population of solutions, each of which is a set of representatives, over a pre-defined number of generations. The best solution of the last generation is chosen to be the set of representatives for the clustering. The SCEC's population consists of the fixed (pre-specified) number of solutions, each of which is a set of representatives. Each solution in the initial population is randomly selected. Populations of the subsequent generations are generated through three genetic operators: mutation, crossover, and copy. SCEC used *K*-tournament selection method (with tournament size of $K = 2$) in selecting potential solutions to participate in creating new population. Different adaptive values are used to control the probabilities of applying each of the three genetic operators to generate new solutions for the subsequent generations.

Apirak Jirayusakul (2007: 23-55) and Apirak Jirayusakul and Surapong Auwatanamongkol (2007: 217-229) proposed two supervised clustering algorithms based on prototype-based clustering methodology: Supervised Growing Neural Gas (SGNG) and Robust Supervised Growing Neural Gas (RSGNG). The SGNG incorporates Growing Neural Gas network with various techniques as Type Two Learning Vector Quantization (LVQ2), adaptive learning rates, and cluster repulsion mechanisms. The SGNG also proposed a new validity based on geometric measurement paradigm in order to determine the optimal number of prototypes. Due to drawbacks of the SGNG of being sensitive to the prototype initialization, the sequence of input data objects, and the presence of noises, the RSGNG is intended to be the robust version of SGNG. The RSGNG incorporates SGNG learning schema with the outlier resistant strategy. Moreover, to determine the optimal number of

prototypes where data objects may include some outliers, a modified validity index was proposed based on the Minimum Description Length technique.

CHAPTER 3

METHODOLOGY

The proposed algorithm is a bottom-up supervised clustering algorithm that relies on many concepts such as grid-based clustering, density-based clustering, and the downward closure property of density used in subspace clustering. The algorithm uses heuristics to partition data spaces into grid cells, then group any adjacent hyper-rectangle grid cells containing majority of data with a same class into a cluster. It can automatically determine the number of intervals to be used to partition each of data dimensions into grid cells clusters, which yields the best clustering in according with a fitness function. The algorithm possesses all of the good clustering properties mentioned in Sheikholeslami, et al. (1998: 428). That is, the algorithm has ability to produce identical results regardless of neither the order of data objects to be processed nor any pre-defined number of clusters. Also, it is able to handle clusters of arbitrary shapes and sizes without making any assumption about the distribution of data objects.

3.1 Definitions

3.1.1 Data Objects

A data object is considered a data point in a *d*-dimensional space. Formally, each data point is a $(d+1)$-tuple in the form $\{a_1, a_2, \dots, a_d, T\}$, where a_i represents the value of the i^{th} predictor variable (or attribute) and T represents the value of the target variable (or class label) of the data point (Ye and Li, 2005: 2-3) .

For instance, $\{1.54, 270, 46.8, A\}$ represents an object which values of the first attribute, the second attribute, and the third attribute are 1.54, 270 and 46.8 respectively, and which class label is "*A*".

3.1.2 Grid Cells

Let $A_1, A_2, \dots, A_d$ be sets of dimensions (or attributes, or predictor variables) of any datasets, and let $A = A_1 \times A_2 \times \dots \times A_d$ be the *d*-dimensional data space. The problem is to divide the data space A into $m = \prod_{i=1}^{d} P_i$ non-overlapping hyper-rectangular grid cells, where P_i represents the number of intervals in the i^{th} dimension of *d*-dimensional data space. A cell is defined by a set of *d*-dimensional hyperplanes, all of which are parallel to $(d-1)$ coordinate axes.

To accomplish this, the range of the value domain of each dimension A_i is partitioned into P_i number of mutually exclusive equal-size right-opened intervals $I_i^j = \lfloor l_i^j, h_i^j \rangle,\ 1 \le j \le P_i$, where l_i^j and h_i^j respectively denotes the start value and end value of the j^{th} interval in the i^{th} dimension, and hence each cell is represented in the form $U = \{I_1, I_2, \dots, I_d\}$ (Agrawal, et al., 1998: 95). A data object $a = \{a_1, a_2, \dots, a_d\}$, where a_i is the value of the i^{th} dimension, is said to lie in the cell U only if $l_i \le a_i < h_i$ for all I_i.

3.1.3 Clusters

As defined by Agrawal, et al. (1998: 96), a cluster is a maximal set of connected dense cells in *d*-dimensions. The problem is to separate all identified dense cells D into $D_1, D_2, \dots, D_k$ sets, such that all cells in the set D_i are said to be connected, and no two cells, $U_i \in D_i,\ U_j \in D_j$ with $i \ne j$ are connected. Two *d*-dimensional cells U_1 and U_2 are declared connected, either in case they share at least one common corner point, or there exists another *d*-dimensional cell U_s to which both U_1 and U_2 are connected.

If a running number is assigned to each interval in all dimensions, starting from 1 to P_i, where P_i is the number of intervals in the i^{th}dimension, each cell can be represented in the form $U_j = \{I_{1j}, I_{2j}, \dots, I_{dj}\}$, where I_{ij} is the interval number of the cell j in the i^{th} dimension. Cells $U_1 = \{I_{11}, I_{21}, \dots, I_{d1}\}$ and $U_2 = \{I_{12}, I_{22}, \dots, I_{d2}\}$ are claimed connected if all $|I_{i1} - I_{i2}| \le 1$, where I_{i1} and I_{i2} are the interval numbers of the i^{th} dimension of U_1 and U_2 respectively.

3.2 Fitness Function

The objective of supervised clustering is to identify groups of data objects, that possess low impurities and few groups as possible. To accomplish this, Zeidat, et al. (2006: 3-4) proposed the following fitness function, $q(x)$, as a validity measurement to evaluate the performance of a supervised clustering algorithm,

$$\boldsymbol{q(x) = Impurity(x) + \beta * Penalty(k)}$$

$$\textit{where} \quad \boldsymbol{Impurity(x) = \frac{number\ of\ minority\ objects}{n}}$$

$$\boldsymbol{Penalty(k) = \begin{cases} \sqrt{\frac{k-c}{n}}, & k > c \\ 0, & k \leq c \end{cases}}$$

where parameter k, c, and n represent the number of generated clusters, the number of classes, and the number of data objects respectively.

The proposed fitness function consists of 2 contradictory parts, $Impurity()$ and $Penalty()$. Due to the objective of supervised clustering, the $q(x)$ value must be kept as low as possible. Further split of data objects into more clusters may cause a decrease on $Impurity()$ value but an increase in $Penalty()$ value. The parameter β puts a weight on the significance of the $Penalty()$ part against the $Impurity()$ part, i.e. the higher the β value, the higher the significance of the $Penalty()$ part. Normally, the β value is chosen between 0 and 5.

Under the consideration that the above fitness function can certainly lead supervised clustering to yield the most effective solution, this $q(x)$ function is chosen to be the fitness function for the proposed algorithm.

3.3 Grid-based Supervised Clustering Algorithm

The proposed algorithm is a bottom-up supervised clustering algorithm relying on the combination of the concepts of grid-based clustering, density-based clustering,

and subspace clustering. The basic idea of the proposed algorithm is to create uniform size grid cells over the whole data space, resulting in partitioning data objects into a number of groups abiding by the region of each cell. Clusters are afterward defined by merging together all connected nonempty cells with the same class labels. In consequence, the data objects lying inside the region of such connected cells are claimed to be in the same cluster. For the proposed algorithm, each dimension is partitioned into same-size intervals, under the condition that the numbers of intervals of different dimensions are allowed to be different.

The key to the success of the algorithm is to use the proper number of intervals for each partitioning of a dimension. The number of intervals for each dimension must be carefully selected so the smallest value of the fitness function $q(x)$ is achieved. To fulfill this, the proposed algorithm comprises 2 steps: Dimension Ordering step and Subspace Clustering step.

3.3.1 Dimension Ordering Step

This step is aimed to be the preparatory part of the next step, with the objective to re-arrange the order of dimensions to be processed sequentially in the subspace clustering procedure. This step determines the order of dimensions to be processed sequentially in the subspace clustering step. The step considers partitioning each dimension independently into a number of equal intervals which yields the smallest fitness function, $q(x)$. A linear search for the number of intervals that produces the smallest fitness $q(x)$ value can be done by starting the search from the number of intervals equal to one, and keeping incrementing the number until there is an obvious increase on the fitness value. Using the linear search method, the number of iterations needed for searching the optimal number of intervals is rather high. To save the processing time, the algorithm still proposes a search for the optimal number of intervals that is based on the gradient descent concept. The Dimension Ordering step comprises two tasks: clustering based on each individual dimension and dimension sequencing.

3.3.1.1 Clustering Based on Each Individual Dimension

Consider that different sets of data have particular characteristic that might result in non-equivalent competence in the clustering activities. The first task of

the Dimension Ordering step is to approximate the clustering potential hidden in each dimension when employed solely in the clustering process. The individual potential for each of a dimension is measured by mean of the value of the possible smallest fitness function, $q(x)$, each specific dimension can produce. To determine the individual potential, a subspace clustering based on each dimension is performed and the fitness function value is evaluated.

With the intention to find the optimal number of intervals (noi) for the dimension partitioning as quickly as possible, the gradient descent method is used in searching for the optimal noi value. During the search, let noi_n represents the current noi value and q_n be the corresponding $q(x)$ value of the noi_n, the new noi value, represented by $noi_{(n+1)}$, is calculated by the following formula,

$$\boldsymbol{noi_{(n+1)} = noi_n - \eta\left[\frac{\Delta q_n}{\Delta noi_n / maxnoi}\right] + momentum}, \quad \textit{if } \boldsymbol{n} > 1$$

$$\boldsymbol{noi_{(0)} = 1} \quad \textit{and} \quad \boldsymbol{noi_{(1)} = 2}$$

where Δq_n represents the difference between the current $q(x)$ value (q_n) and its previous value $(q_{(n-1)})$, as well as Δnoi_n represents the difference between the current noi value, noi_n, and its previous value, $noi_{(n-1)}$.

Since the $q(x)$ values are certainly less than 1.0, whereas the noi values are higher than 1, the value of Δnoi_n is normalized by a maximum value of noi, i.e. $maxnoi$, in order to correctly determine the gradient or the ratio between Δq_n and Δnoi_n. The learning rate of the gradient descent formula is represented by the symbol η. The momentum term is employed to give weight on the current increment/decrement of the noi in order to avoid the convergence to local optima. The term is formularized as

$$\Upsilon\,(|\Delta \boldsymbol{noi_n}|)$$

where Υ is the momentum weight of the momentum term.

The noi value calculated from the formula is always round to an integer value. When the noi value gets incremented, the noi value is always round up in

order to facilitate the forward move during early stages of the search. However, when it is decremented, it is round to the nearest integer.

The gradient descent search for the optimal noi value for each individual dimension clustering is terminated when either of the two conditions occurs two times consecutively. We assume when such event happens the optima has been reached, therefore, the search should be terminated. The first condition is that there is no decrement of $q(x)$ value. The second condition is when the recommended noi results in deficient decrement in the $q(x)$ values. This condition is measured by dividing the absolute value of the latest slope, computed as $\left|\frac{q_{(n)}-q_{(n-1)}}{noi_{(n)}-noi_{(n-1)}}\right|$, by the absolute value of the initial slope, computed as $\left|\frac{q_{(2)}-q_{(1)}}{noi_{(2)}-noi_{(1)}}\right|$. In case the result value is less than 0.001, the algorithm claims that such decrease of the $q(x)$ value is not worth-while the increase in the noi value.

In addition, when the search goes backward, i.e. noi is decremented, the current value of noi and the new noi (after decremented from the current noi) will mark the range of noi to be searched. If the search moves current noi value beyond the range, the search is also terminated. This is similar to the movement of a pendulum which can swing only back and forth with gradually reduced range to swing.

3.3.1.2 Dimension Sequencing Task

Abide by the observation, the dimensions possessing lower $q(x)$ values when working individually have tendency to yield better result when working mutually in the subspace clustering. Using greedy approach, such dimensions should be given higher priority in being used early in the subspace clustering process. Hence, the last task of the Dimension Ordering step is to sort the dimensions into a list in ascending order on their smallest $q(x)$ values achieved from the first task. The list will be used to guide the subspace clustering in the next step.

3.3.2 Subspace Clustering Step

The intention of this step is to find out the delineation of grid cells that would produce clustering with the possible smallest $q(x)$ value when all dimensions are

considered together. The Subspace Clustering step comprises two tasks: Grid Cell Creation task and Cluster Formation task.

3.3.2.1 Grid Cells Creation Task

In this task, grid cells are created in bottom up fashion by gradually and repeatedly partition data space using one additional dimension at a time. Using the heuristic mentioned in section 3.3.1.1, the partitioning performs on the dimensions in sequence based on the order list created by the first step. The partitioning starts from creating the first-level grid cells using the first dimension in the list together with the optimal number of intervals for the dimension previously derived in the first step. Only data objects residing in each of the first-level grid cells as well as the grid cell information are then written into the external file, cell by cell. Next, the nonempty first-level grid cells, retrieved from the external file, will be partitioned using the second dimension in the list.

Since the grid space has already been partitioned by the first dimension, the optimal number of intervals for the second dimension can be different from the one derived in the first step. Hence, the optimal number of intervals for the second dimension must be derived again using the same gradient descent method as in the first step, except that clusters are formed using the first and the second dimensions and the number of intervals of the first dimension is fixed. The process then continues on the third dimension and so on until all dimensions have been partitioned.

Refer to the fact that the number of generated grid cells can be computed as $\prod_{i=1}^{d} P_i$ where d represents the number of dimensions and P_i represents the number of intervals in the i^{th} dimensions of d-dimensional space, the number of created cells increases dramatically whenever the number of dimensions increases. When the number of dimensions is large, not all grid cells contain data objects, and the number of grid cells containing data objects is usually tremendously small when compared with the number of created cells.

As only nonempty cells in the current dimensional-level grid space are kept for the processing of the subsequent higher dimensional level, this procedure results in saving a lot of processing time since large parts of search space are discarded. As a consequence, the proposed algorithm allows only $(d-1)$

dimensional nonempty grid cells to be candidates for the generation of *d*-dimensional nonempty grid cells.

3.3.2.2 Cluster Formation Task

To create final clusters (or partial clusters formed during the first or the second step), connected nonempty same-class labeled cells are merged into a same cluster. The input for the cluster formation task is a set of cell blocks D, each of which consists of cell's information, cell's class label, and data objects belonging to that cell. Starting from any cell $U \in D$ as the seed cell to form a cluster, the task searches in D to find for all $U_j \in D$, which are connected with U, or any members of U, and have the same class label as U. All U_js are then put into a same cluster as well as removed from D. The task arbitrary selects the next seed cell U from D. It then performs the same process to form the next cluster. The iterative process stops when all cells have been removed from D.

With this cluster formation procedure, the proposed algorithm can generate clusters of any shapes and sizes without presuming any specific mathematical form for data distribution, and can produce identical results regardless of the order in which input data objects are presented.

CHAPTER 4

EXPERIMENTAL RESULTS

The objective of this chapter is to illustrate the experiments designed to evaluate the effectiveness of the proposed algorithm. Two sets of experiments were performed. The first set was performed on datasets obtained from University of California at Irving Machine Learning repository. The experimental results are compared to those reported in Zeidat et al. (2006: 14-15) and Apirak Jirayusakul (2007: 96). The second one was performed on two-dimension synthetic datasets under the permission of the author of Apirak Jirayusakul (2007: 61-62, 66, 69-70, 73-74).

4.1 The Experiments on UCI Datasets

The objective of this experiment is to evaluate the performance of the proposed algorithm in the comparative manner, with the other supervised clustering algorithms. The experiments were performed on four datasets obtained from University of California at Irving Machine Learning repository: *Iris-Plans*, *Pimma-Indian Diabetes*, *Vehicle Silhouettes*, and *Image-Segmentation,* using the following parameters: $\beta = 0.1,\ \eta = 0.5,\ maxnoi = 100,\ \Upsilon = 0.3$. The properties of the four datasets are shown in Table 4.1. The results in term of the fitness function values $q(x)$ from the experiments are compared with those results from SPAM, SREDHCR, and SCEC reported in Zeidat et al. (2006: 14-15), and the best solutions from SGNG and RSGNG in Apirak Jirayusakul (2007: 96).

Table 4.1 List of the Properties of UCI Datasets Used in the Experiments

Dataset Name	No. of Examples	No. of Attributes	No. of Classes
Iris-Plants	150	4	3
Pima-Indian Diabetes	768	8	2
Vehicle Silhouettes	846	18	4
Image-Segmentation	2100	19	7

4.1.1 Experimental Results from Dimension Ordering Step

The task of the Dimension Ordering step is to find out the potential $q(x)$ value each attribute (or dimension) could give when each is considered alone for the grid space partitioning. The attributes are then ordered in the ascending sequence based on their $q(x)$ values. Table 4.2 (a)-(d) and Table 4.3 (a)-(d) show the ordering results of the attributes for the four datasets, when β is set at 0.1 and 0.4 respectively.

Table 4.2 Results from Dimension Ordering Step at $\beta = 0.1$

(a) Iris-Plants [on β=0.1]

Oder No.	Attribute No.	*NOI*	*Q(x)* Value	Oder No.	Attribute No.	*NOI*	*Q(x)* Value
1	4	3	0.04000	3	1	4	0.26488
2	3	3	0.04667	4	2	2	0.42000

(b) Pima-Indian Diabetes [on β=0.1]

Oder No.	Attribute No.	*NOI*	*Q(x)* Value	Oder No.	Attribute No.	*NOI*	*Q(x)* Value
1	2	7	0.25260	5	7	7	0.33713
2	8	3	0.32783	6	5	2	0.34115
3	6	3	0.33333	7	4	3	0.34766
4	1	2	0.33594	8	3	1	0.34896

(c) Vehicle Silhouettes [on β=0.1]

Oder No.	Attribute No.	*NOI*	*Q(x)* Value	Oder No.	Attribute No.	*NOI*	*Q(x)* Value
1	7	14	0.46432	10	4	11	0.58406
2	8	17	0.46868	11	10	8	0.58524
3	9	12	0.47614	12	18	14	0.58642
4	12	7	0.51171	13	17	10	0.58870
5	11	11	0.51786	14	13	11	0.59907
6	3	12	0.53205	15	14	13	0.61243
7	1	12	0.55308	16	15	9	0.66194
8	2	10	0.58161	17	16	8	0.66325
9	6	10	0.58397	18	5	6	0.68203

(d) Image-Segmentation [on β=0.1]

Oder No.	Attribute No.	*NOI*	*Q(x)* Value	Oder No.	Attribute No.	*NOI*	*Q(x)* Value
1	11	16	0.40742	11	14	15	0.50762
2	10	14	0.43456	12	8	26	0.71361
3	12	14	0.44000	13	6	23	0.71631
4	17	18	0.44381	14	1	8	0.76952
5	13	16	0.44429	15	4	4	0.83143
6	18	15	0.44547	16	5	3	0.84190
7	19	14	0.44905	17	7	6	0.85000
8	16	15	0.46714	18	9	4	0.85333
9	15	16	0.46714	19	3	1	0.85714
10	2	10	0.49524				

Table 4.3 Results from Dimension Ordering Step at $\beta = 0.4$

(a) Iris-Plants [on β=0.4]

Oder No.	Attribute No.	*NOI*	*Q(x)* Value	Oder No.	Attribute No.	*NOI*	*Q(x)* Value
1	4	3	0.04000	3	1	4	0.29952
2	3	3	0.04667	4	2	2	0.42000

(b) Pima-Indian Diabetes [on β=0.4]

Oder No.	Attribute No.	*NOI*	*Q(x)* Value	Oder No.	Attribute No.	*NOI*	*Q(x)* Value
1	2	7	0.25260	5	5	2	0.34115
2	6	3	0.33333	6	7	2	0.34766
3	1	2	0.33594	7	4	3	0.34766
4	8	3	0.33865	8	3	1	0.34896

(c) Vehicle Silhouettes [on β=0.4]

Oder No.	Attribute No.	*NOI*	*Q(x)* Value	Oder No.	Attribute No.	*NOI*	*Q(x)* Value
1	7	14	0.48495	10	6	10	0.60183
2	8	17	0.49174	11	17	10	0.60656
3	9	12	0.49677	12	18	8	0.60875
4	12	7	0.52203	13	4	4	0.61229
5	11	11	0.53245	14	13	11	0.61970
6	3	12	0.54664	15	14	6	0.63357
7	1	12	0.56340	16	15	9	0.66194
8	10	8	0.59983	17	16	2	0.67967
9	2	6	0.60004	18	5	6	0.68203

(d) Image-Segmentation [on β=0.4]

Oder No.	Attribute No.	*NOI*	*Q(x)* Value	Oder No.	Attribute No.	*NOI*	*Q(x)* Value
1	11	16	0.41397	11	14	15	0.50762
2	12	14	0.44000	12	8	25	0.71857
3	10	14	0.44111	13	6	18	0.73397
4	17	18	0.44381	14	1	8	0.76952
5	13	16	0.44429	15	4	4	0.83143
6	19	14	0.44905	16	5	3	0.84190
7	18	15	0.45473	17	7	6	0.85000
8	15	16	0.46714	18	9	4	0.85333
9	16	19	0.47016	19	3	1	0.85714
10	2	10	0.49524				

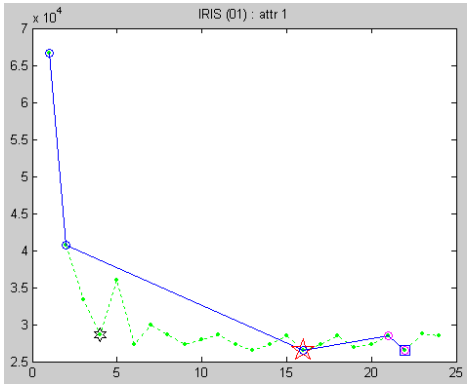
The proposed algorithm proposes the methodology to reduce the execution time by neglecting the processing on the possibly trivial number of intervals (noi) values and going directly to the worthy *noi* values. The methodology has been explained in section 3.3.1. Figure 4.1 (a)-(d) illustrates the searching chain in acquiring the prospective *noi* values performing on each individual attribute of the *Iris-Plants* dataset. Figure 4.2 (a)-(h) are for the *Pima-Indian Diabetes* dataset, Figure 4.3 (a)-(r) are for the *Vehicle Silhouettes* dataset, and Figure 4.4 (a)-(s) are for the *Image-Segmentation* dataset, at β=0.1. Figure 4.5 (a)-(d), Figure 4.6 (a)-(h), Figure

4.7 (a)-(r), and Figure 4.8(a)-(s) show the results from the same scenarios, but at β=0.4. Values on the x-axis in the figures represent the noi values, and values on the y-axis represent the $q(x)$ values achieved for the given noi. The meanings of the symbols used in the figures are described in Table 4.4.

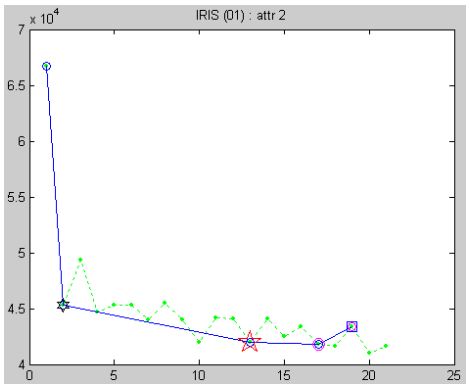
As can be seen from the figures, the $q(x)$ value achieved from the corresponding prospective noi value is gradually decreased by the gradient descent where the current gradient is negative. However, it can pass through the optimal value where $q(x)$ is minimal and the noi value is as small as possible. The noi value can get increased as the current gradient becomes positive and eventually settled down at the optimal value where the current gradient becomes almost zero.

Table 4.4 The Symbols and their Meanings Used in Figure 4.1 thru 4.8

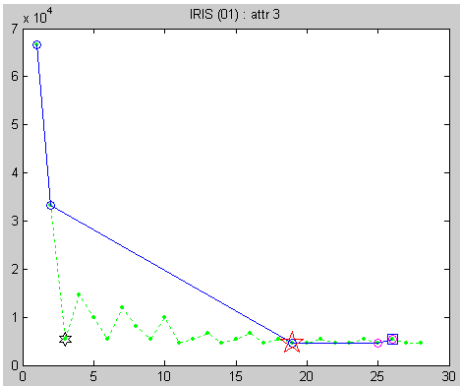
Symbol	Symbol Meanings
green point	the skipped ***noi*** value
blue circle	the prospective ***noi*** value that generates the lower ***q(x)*** value than the previous value
magenta circle	the prospective ***noi*** value that generates the higher ***q(x)*** value than the previous value
yellow circle	the swinging range
red five-pointed star	the final result ***noi*** value
black six-pointed star	the ***q(x)*** value that might proposed by the ordinary searching procedure (Figure 3.2)
blue square	the termination point
green dotted line	the linkage between every value of ***noi***
yellow dotted line	The linkage between the swung ***noi*** values
blue solid line	the linkage between the prospective ***noi*** values



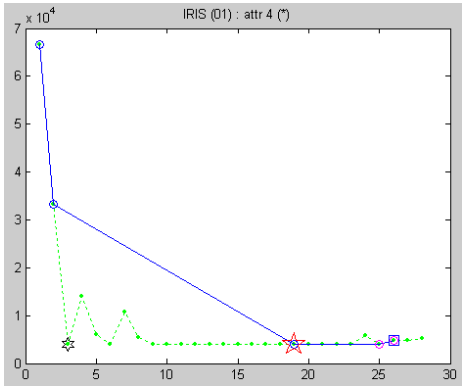
(a) ***NOI*** Searching Chain on Attr. No. 1



(b) ***NOI*** Searching Chain on Attr. No. 2

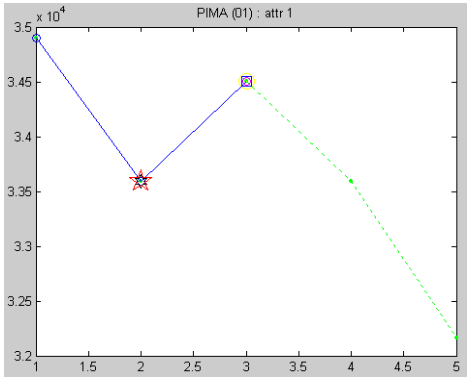


(c) ***NOI*** Searching Chain on Attr. No. 3

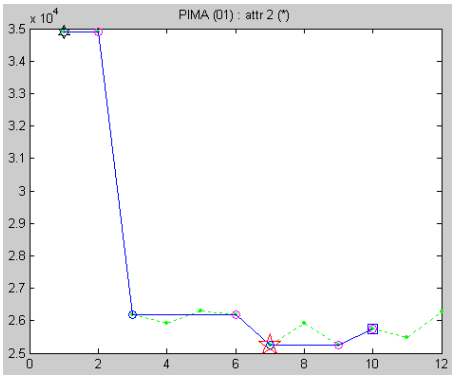


(d) ***NOI*** Searching Chain on Attr. No. 4

Figure 4.1 Searching Chain for *NOI* Values on *Iris-Plants* at $\beta = 0.1$

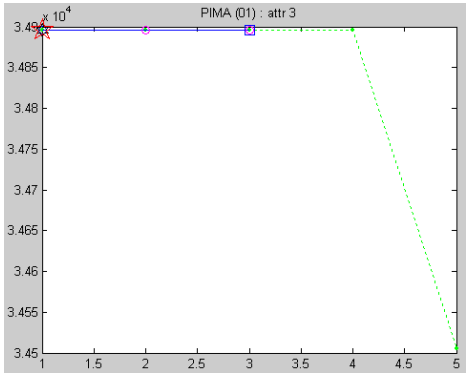


(a) ***NOI*** Searching Chain on Attr. No. 1

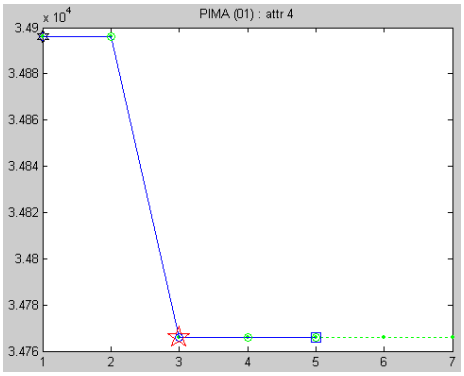


(b) ***NOI*** Searching Chain on Attr. No. 2

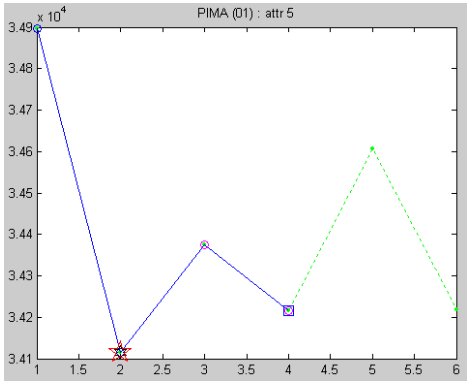
Figure 4.2 Searching Chain for *NOI* Values on *Pima-Indian Diabetes* at $\beta = 0.1$



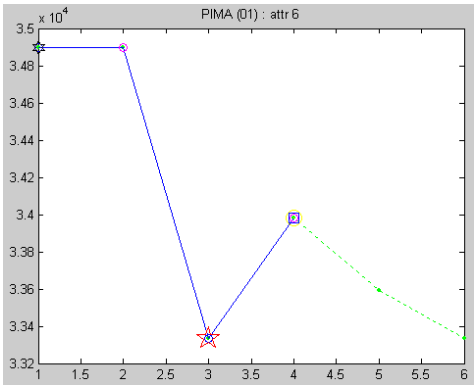
(c) ***NOI*** Searching Chain on Attr. No. 3



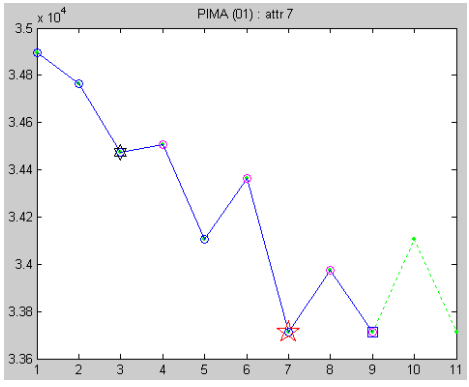
(d) ***NOI*** Searching Chain on Attr. No. 4



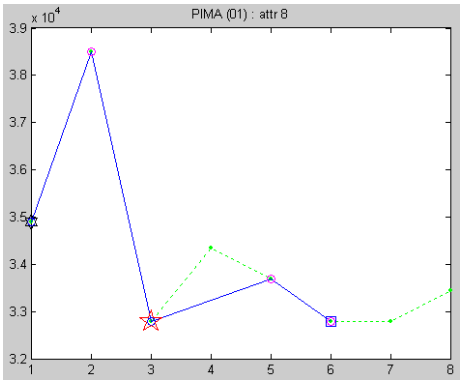
(e) ***NOI*** Searching Chain on Attr. No. 5



(f) ***NOI*** Searching Chain on Attr. No. 6

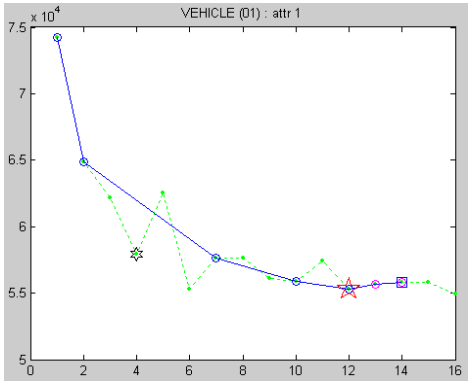


(g) ***NOI*** Searching Chain on Attr. No. 7

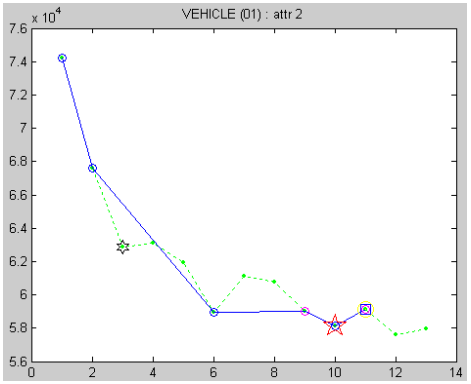


(h) ***NOI*** Searching Chain on Attr. No. 8

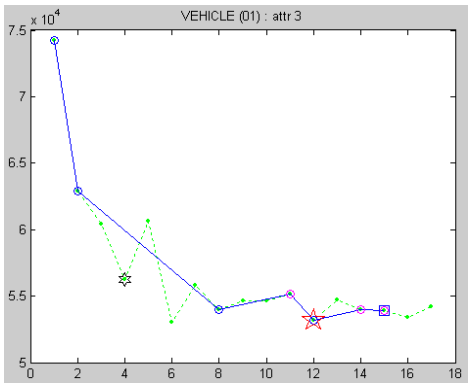
Figure 4.2 Searching Chain for *NOI* Values on *Pima-Indian Diabetes* at $\beta = 0.1$ (Continued)



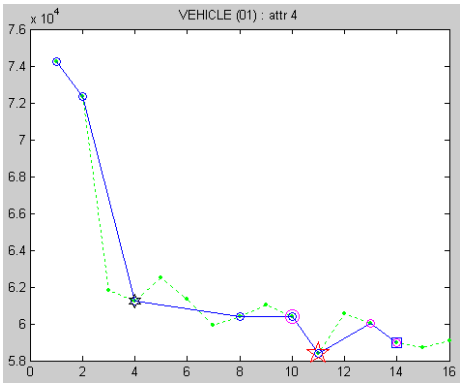
(a) ***NOI*** Searching Chain on Attr. No. 1



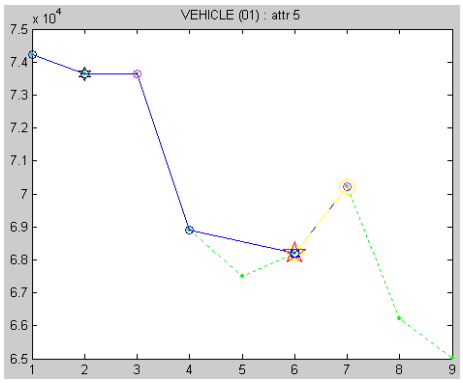
(b) ***NOI*** Searching Chain on Attr. No. 2



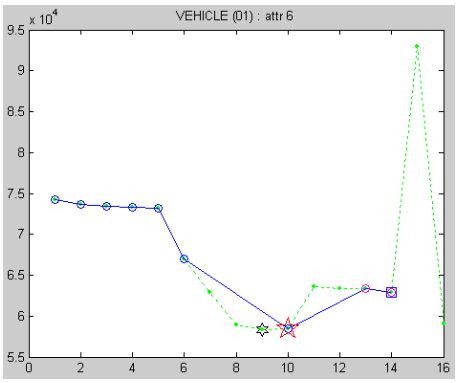
(c) ***NOI*** Searching Chain on Attr. No. 3



(d) ***NOI*** Searching Chain on Attr. No. 4

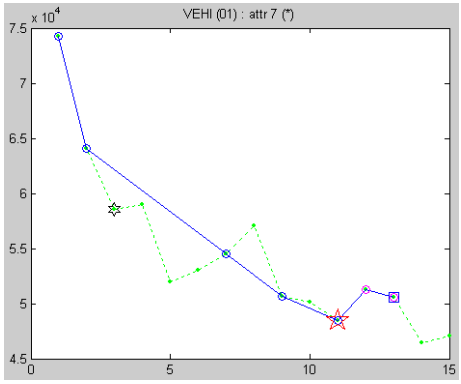


(e) ***NOI*** Searching Chain on Attr. No. 5

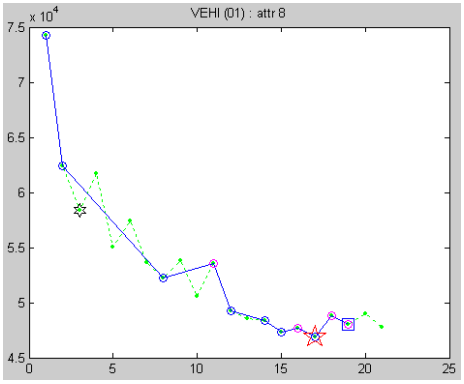


(f) ***NOI*** Searching Chain on Attr. No. 6

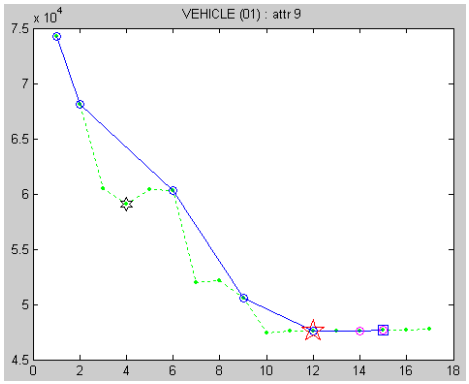
Figure 4.3 Searching Chain for *NOI* Values on *Vehicle Silhouettes* at $\beta = 0.1$



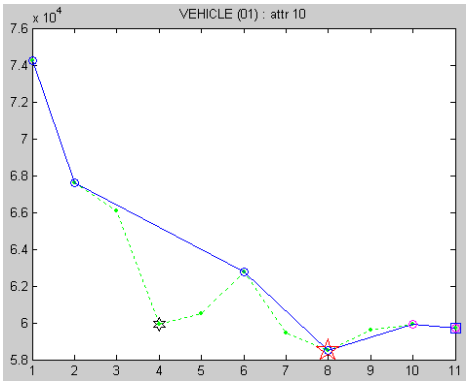
(g) ***NOI*** Searching Chain on Attr. No. 7



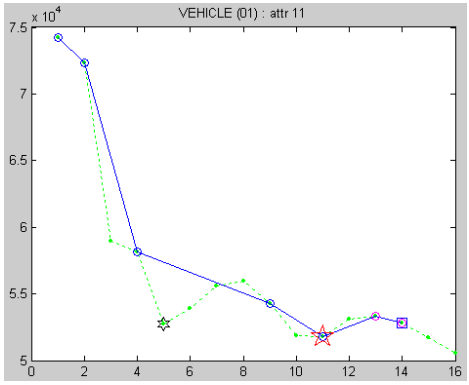
(h) ***NOI*** Searching Chain on Attr. No. 8



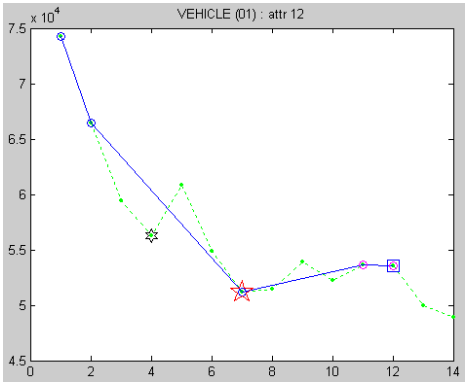
(i) ***NOI*** Searching Chain on Attr. No. 9



(j) ***NOI*** Searching Chain on Attr. No. 10

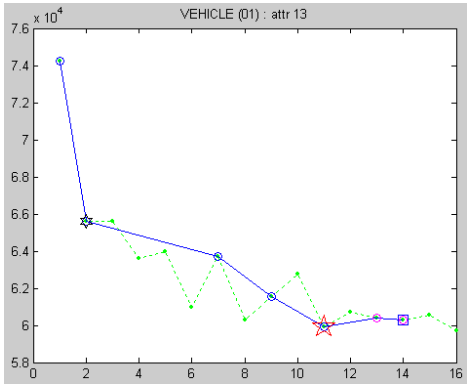


(k) ***NOI*** Searching Chain on Attr. No. 11

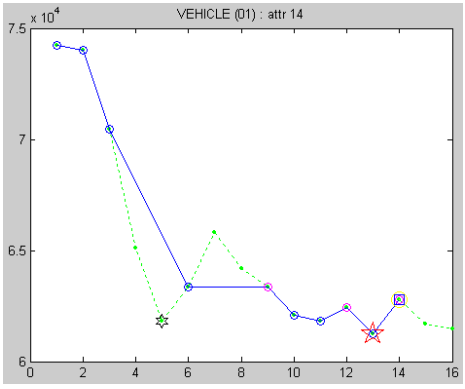


(l) ***NOI*** Searching Chain on Attr. No. 12

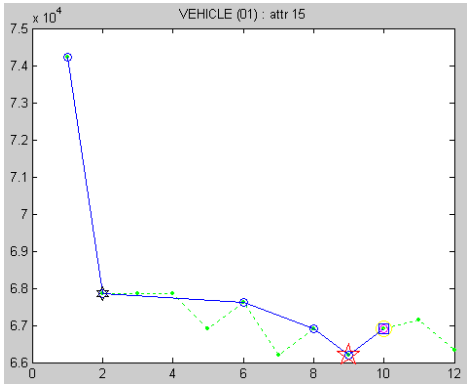
Figure 4.3 Searching Chain for *NOI* Values on *Vehicle Silhouettes* at $\beta = 0.1$ (Continued)



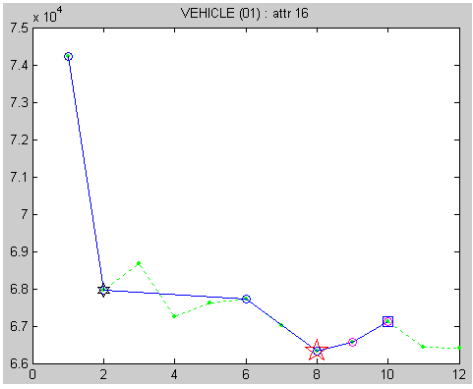
(m) ***NOI*** Searching Chain on Attr. No. 13



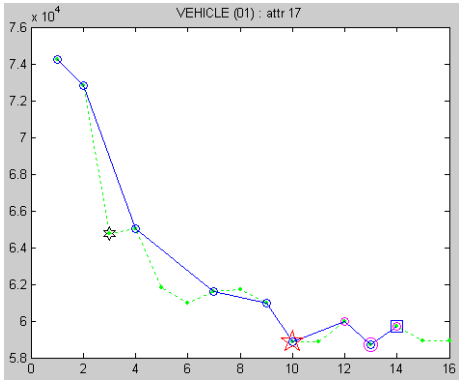
(n) ***NOI*** Searching Chain on Attr. No. 14



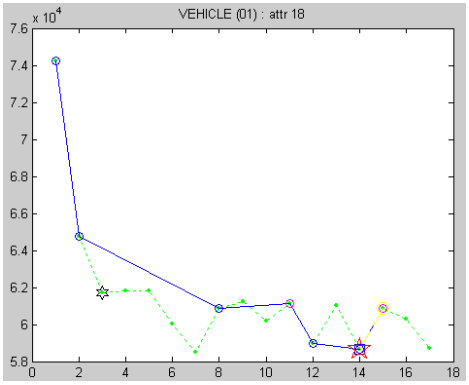
(o) ***NOI*** Searching Chain on Attr. No. 15



(p) ***NOI*** Searching Chain on Attr. No. 16

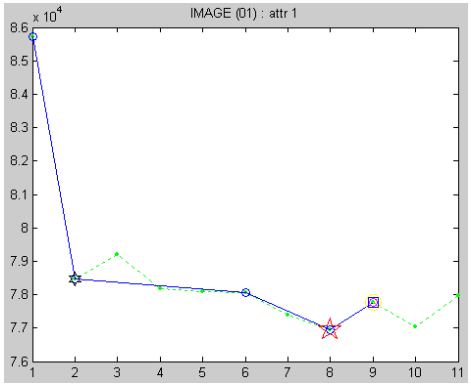


(q) ***NOI*** Searching Chain on Attr. No. 17

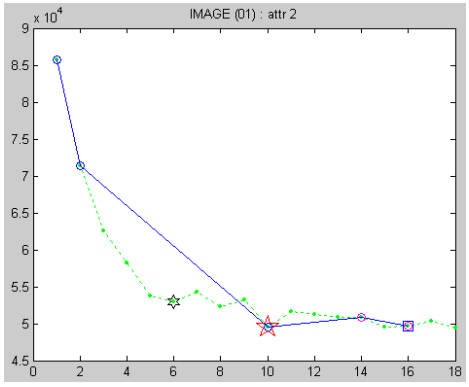


(r) ***NOI*** Searching Chain on Attr. No. 18

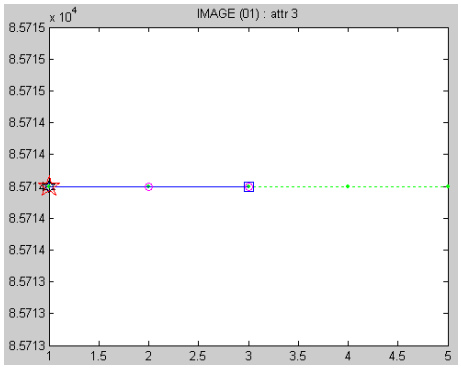
Figure 4.3 Searching Chain for *NOI* Values on *Vehicle Silhouettes* at $\beta = 0.1$ (Continued)



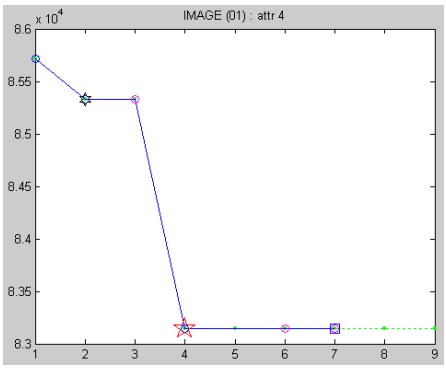
(a) ***NOI*** Searching Chain on Attr. No. 1



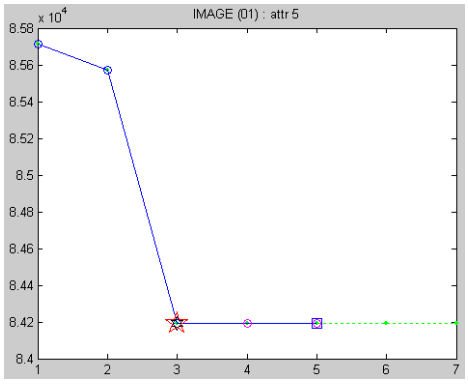
(b) ***NOI*** Searching Chain on Attr. No. 2



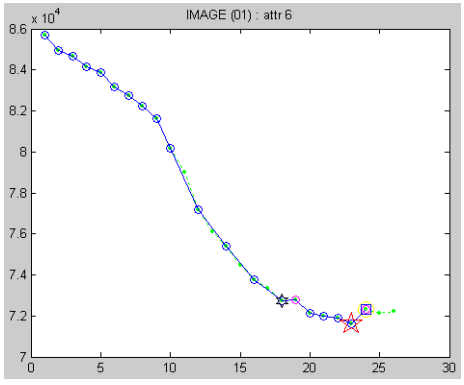
(c) ***NOI*** Searching Chain on Attr. No. 3



(d) ***NOI*** Searching Chain on Attr. No. 4

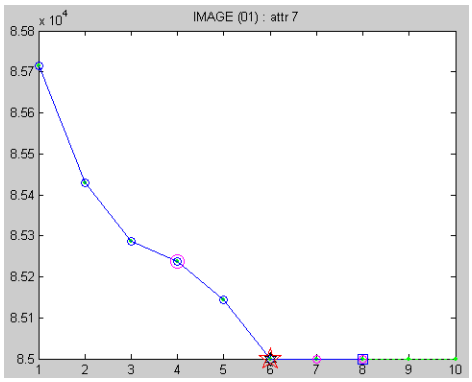


(e) ***NOI*** Searching Chain on Attr. No. 5

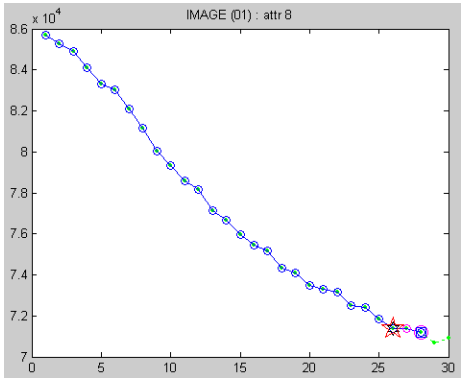


(f) ***NOI*** Searching Chain on Attr. No. 6

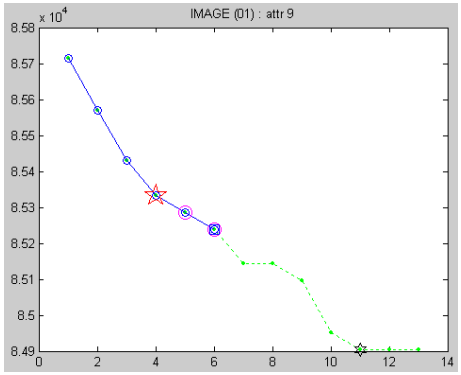
Figure 4.4 Searching Chain for *NOI* Values on *Image-Segmentation* at $\beta = 0.1$



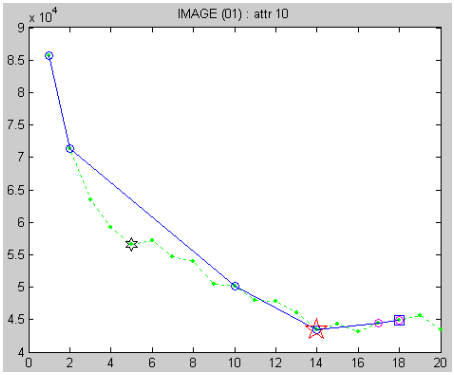
(g) ***NOI*** Searching Chain on Attr. No. 7



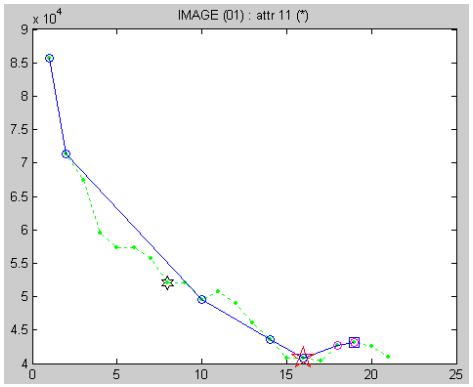
(h) ***NOI*** Searching Chain on Attr. No. 8



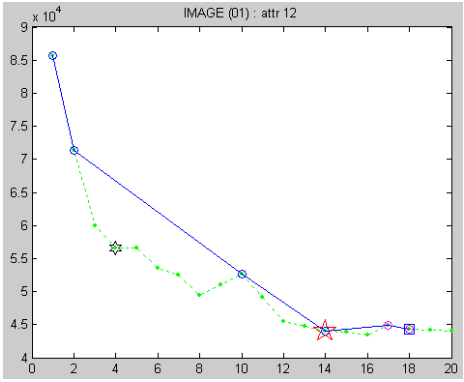
(i) ***NOI*** Searching Chain on Attr. No. 9



(j) ***NOI*** Searching Chain on Attr. No. 10

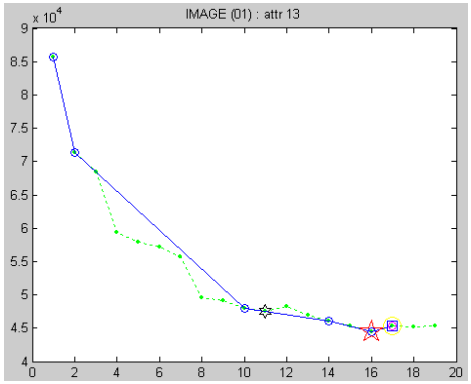


(k) ***NOI*** Searching Chain on Attr. No. 11

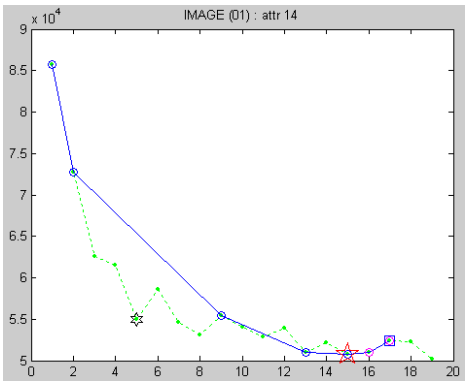


(l) ***NOI*** Searching Chain on Attr. No. 12

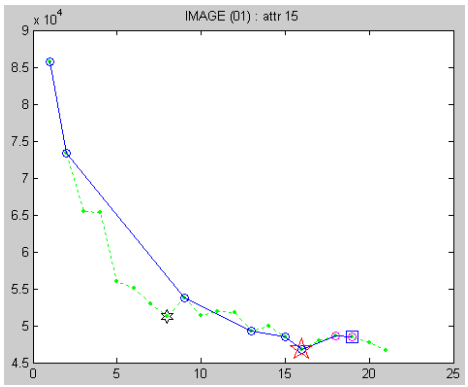
Figure 4.4 Searching Chain for *NOI* Values on *Image-Segmentation* at $\beta = 0.1$ (Continued)



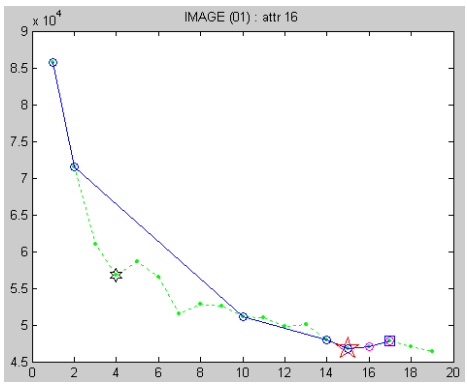
(m) ***NOI*** Searching Chain on Attr. No. 13



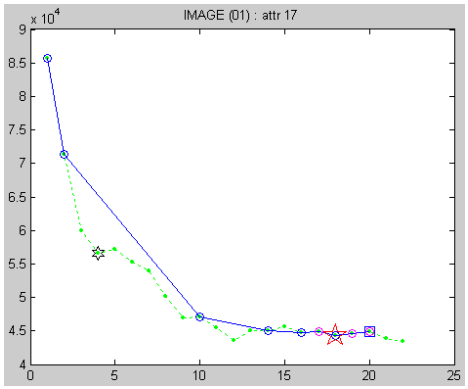
(n) ***NOI*** Searching Chain on Attr. No. 14



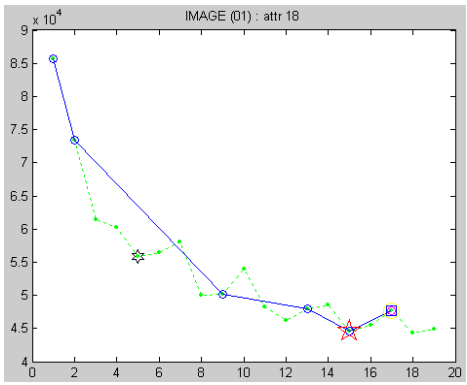
(o) ***NOI*** Searching Chain on Attr. No. 15



(p) ***NOI*** Searching Chain on Attr. No. 16

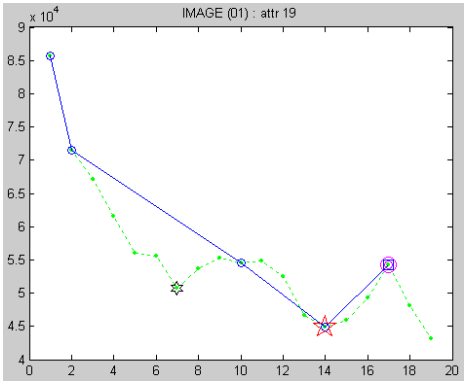


(q) ***NOI*** Searching Chain on Attr. No. 17



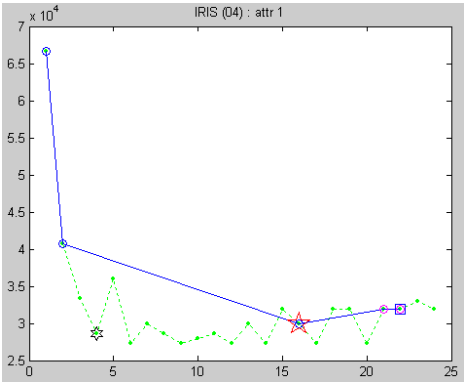
(r) ***NOI*** Searching Chain on Attr. No. 18

Figure 4.4 Searching Chain for *NOI* Values on *Image-Segmentation* at $\beta = 0.1$ (Continued)

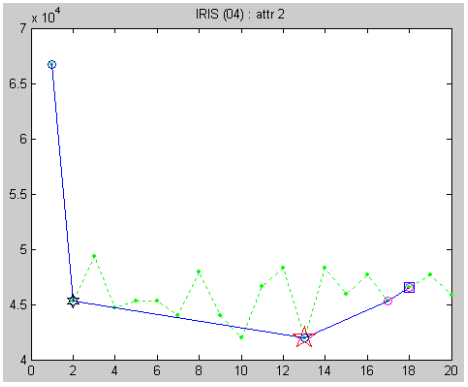


(s) ***NOI*** Searching Chain on Attr. No. 19

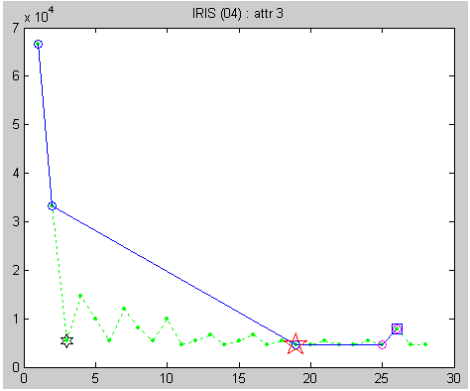
Figure 4.4 Searching Chain for *NOI* Values on *Image-Segmentation* at $\beta = 0.1$ (Continued)



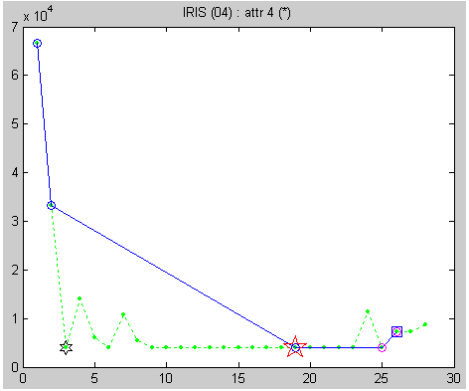
(a) ***NOI*** Searching Chain on Attr. No. 1



(b) ***NOI*** Searching Chain on Attr. No. 2

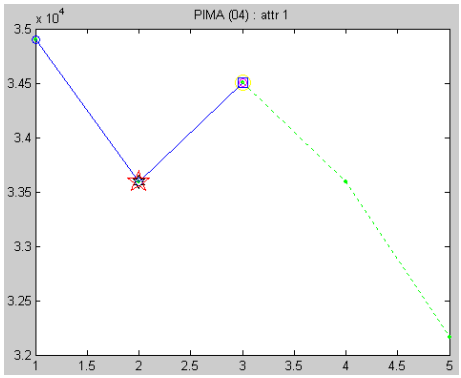


(c) ***NOI*** Searching Chain on Attr. No. 3

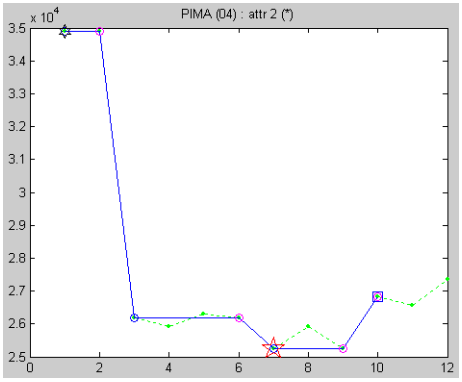


(d) ***NOI*** Searching Chain on Attr. No. 4

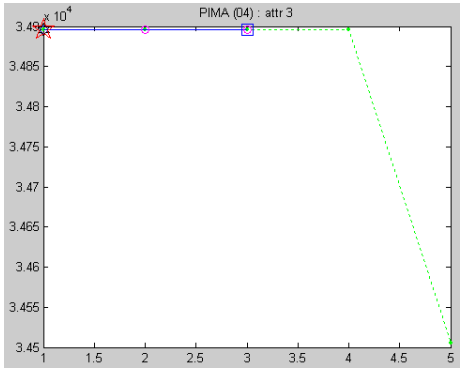
Figure 4.5 Searching Chain for *NOI* Values on *Iris-Plants* at $\beta = 0.4$



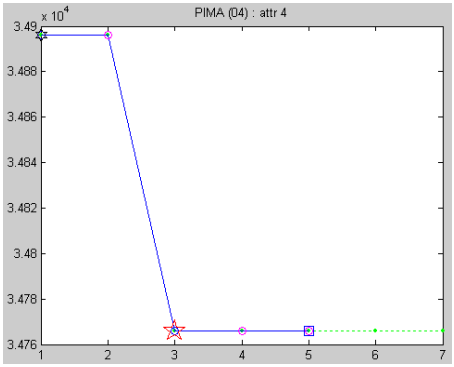
(a) ***NOI*** Searching Chain on Attr. No. 1



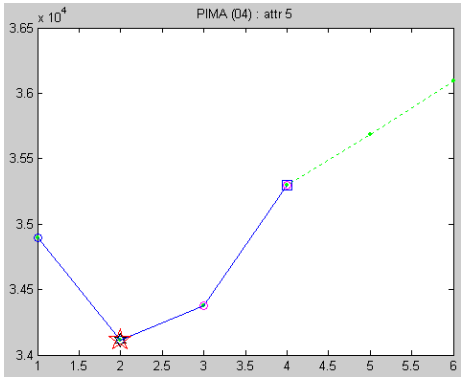
(b) ***NOI*** Searching Chain on Attr. No. 2



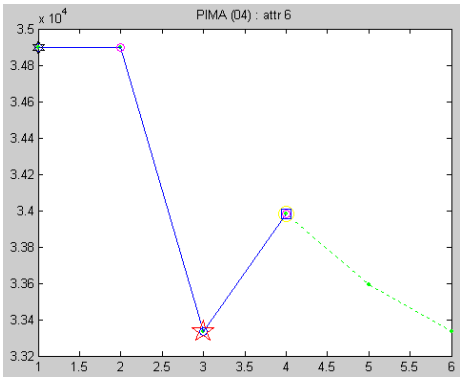
(c) ***NOI*** Searching Chain on Attr. No. 3



(d) ***NOI*** Searching Chain on Attr. No. 4

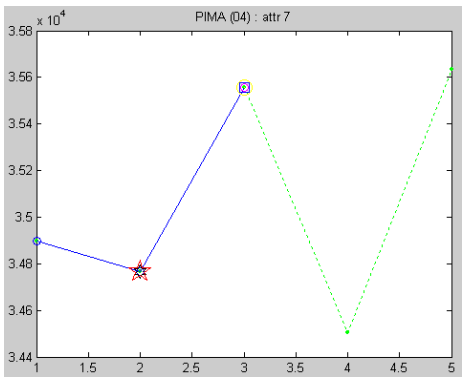


(e) ***NOI*** Searching Chain on Attr. No. 5

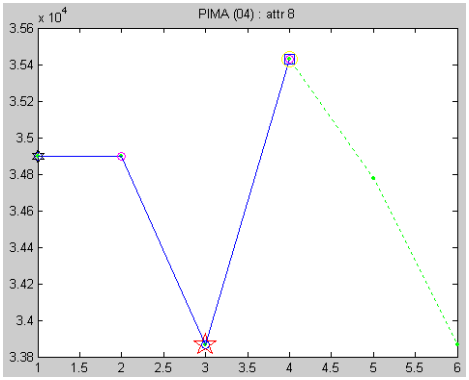


(f) ***NOI*** Searching Chain on Attr. No. 6

Figure 4.6 Searching Chain for *NOI* Values on *Pima-Indian Diabetes* at $\beta = 0.4$

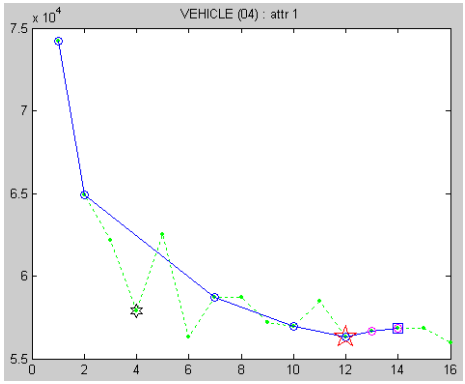


(g) ***NOI*** Searching Chain on Attr. No. 7

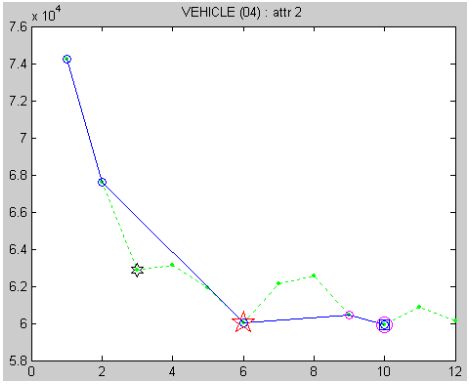


(h) ***NOI*** Searching Chain on Attr. No. 8

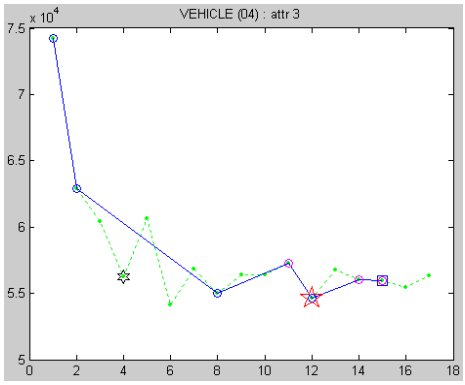
Figure 4.6 Searching Chain for *NOI* Values on *Pima-Indian Diabetes* at $\beta = 0.4$ (Continued)



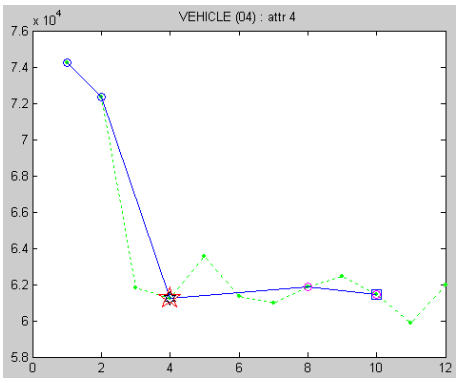
(a) ***NOI*** Searching Chain on Attr. No. 1



(b) ***NOI*** Searching Chain on Attr. No. 2

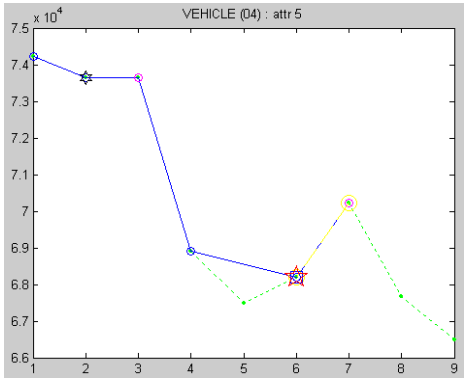


(c) ***NOI*** Searching Chain on Attr. No. 3

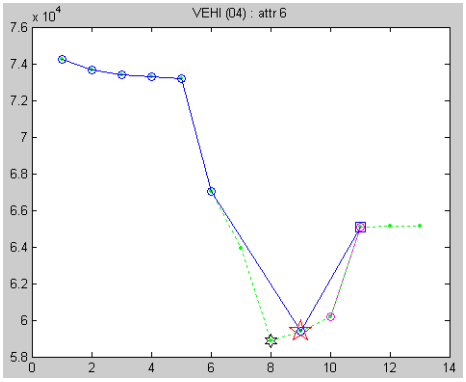


(d) ***NOI*** Searching Chain on Attr. No. 4

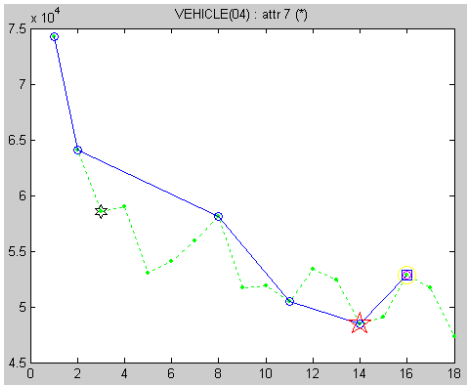
Figure 4.7 Searching Chain for *NOI* Values on *Vehicle Silhouettes* at $\beta = 0.4$



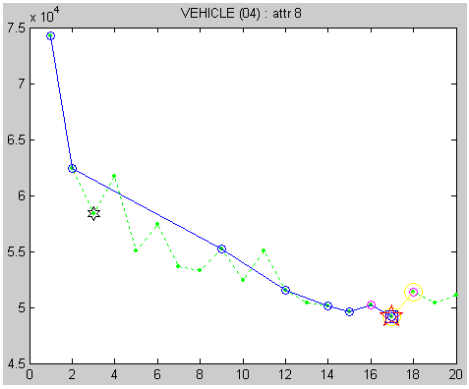
(e) ***NOI*** Searching Chain on Attr. No. 5



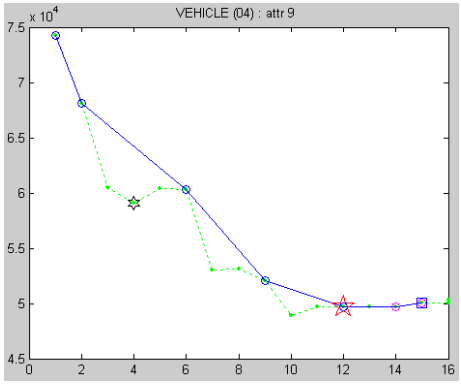
(f) ***NOI*** Searching Chain on Attr. No. 6



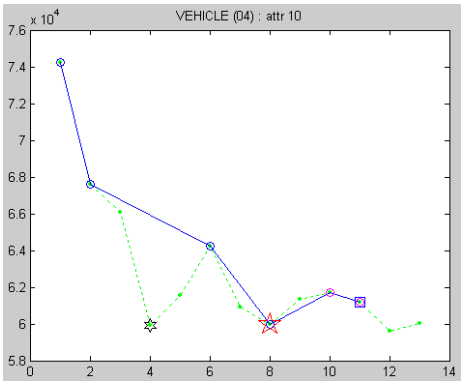
(g) ***NOI*** Searching Chain on Attr. No. 7



(h) ***NOI*** Searching Chain on Attr. No. 8

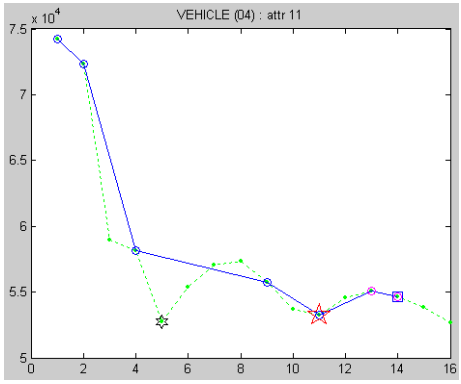


(i) ***NOI*** Searching Chain on Attr. No. 9

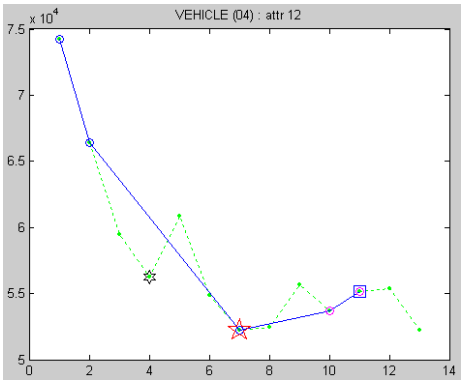


(j) ***NOI*** Searching Chain on Attr. No. 10

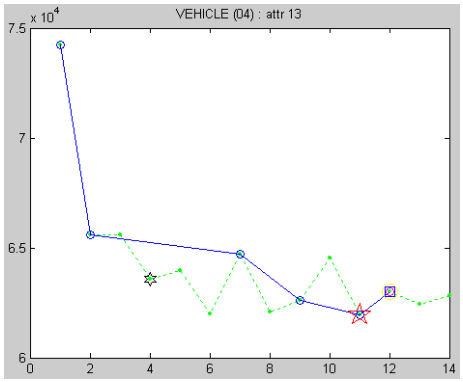
Figure 4.7 Searching Chain for *NOI* Values on *Vehicle Silhouettes* at $\beta = 0.4$ (Continued)



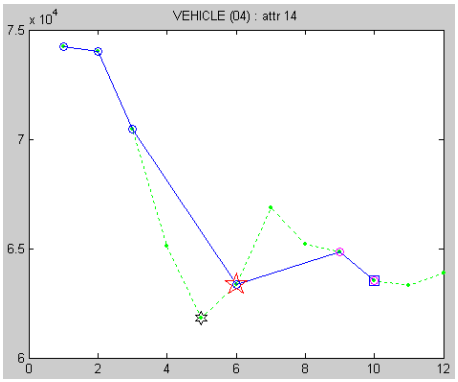
(k) ***NOI*** Searching Chain on Attr. No. 11



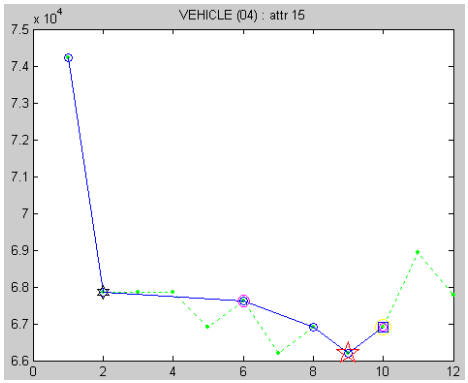
(l) ***NOI*** Searching Chain on Attr. No. 12



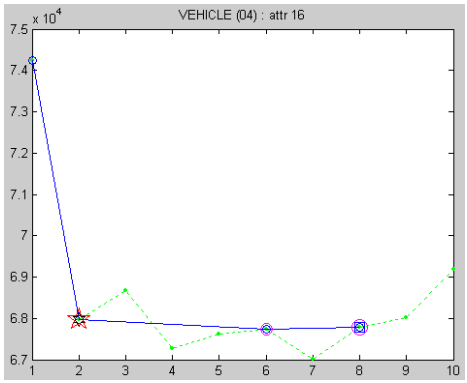
(m) ***NOI*** Searching Chain on Attr. No. 13



(n) ***NOI*** Searching Chain on Attr. No. 14

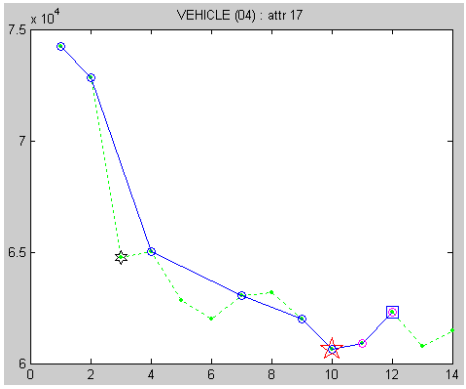


(o) ***NOI*** Searching Chain on Attr. No. 15

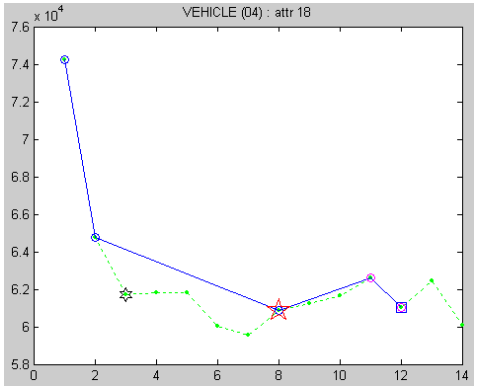


(p) ***NOI*** Searching Chain on Attr. No. 16

Figure 4.7 Searching Chain for *NOI* Values on *Vehicle Silhouettes* at $\beta = 0.4$ (Continued)

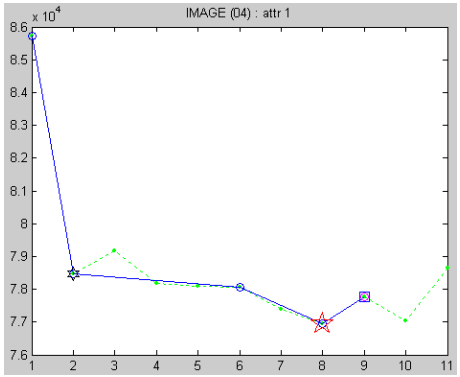


(q) ***NOI*** Searching Chain on Attr. No. 17

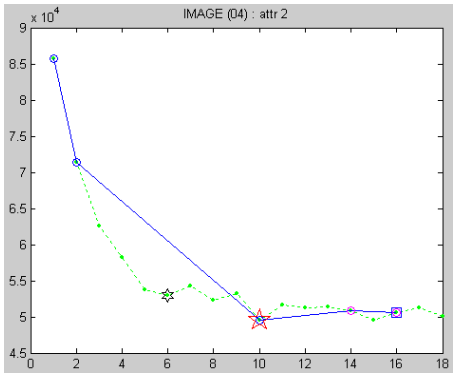


(r) ***NOI*** Searching Chain on Attr. No. 18

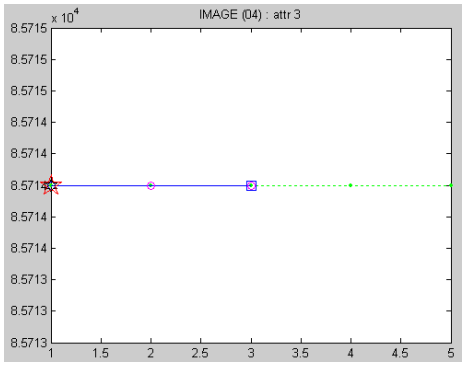
Figure 4.7 Searching Chain for *NOI* Values on *Vehicle Silhouettes* at $\beta = 0.4$ (Continued)



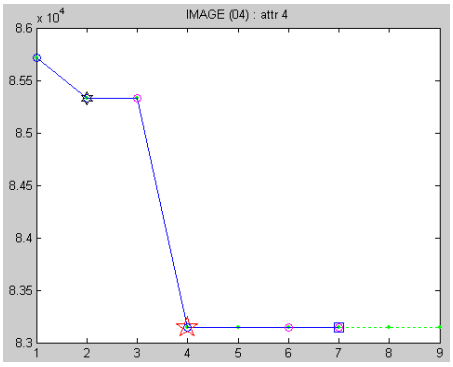
(a) ***NOI*** Searching Chain on Attr. No. 1



(b) ***NOI*** Searching Chain on Attr. No. 2

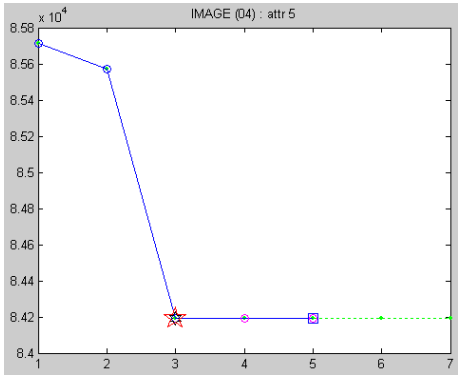


(c) ***NOI*** Searching Chain on Attr. No. 3

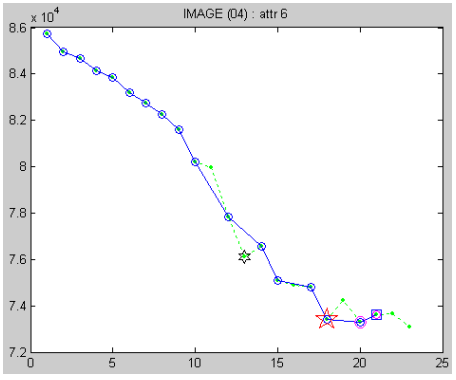


(d) ***NOI*** Searching Chain on Attr. No. 4

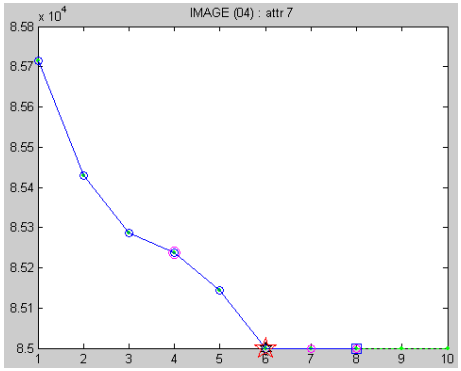
Figure 4.8 Searching Chain for *NOI* Values on *Image-Segmentation* at $\beta = 0.4$



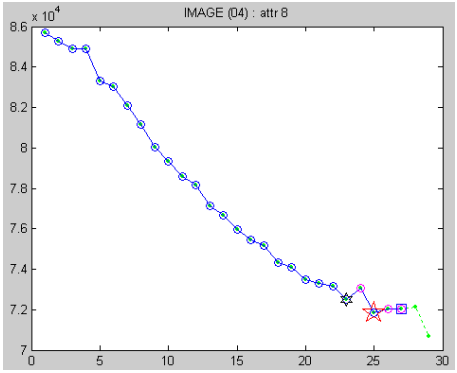
(e) ***NOI*** Searching Chain on Attr. No. 5



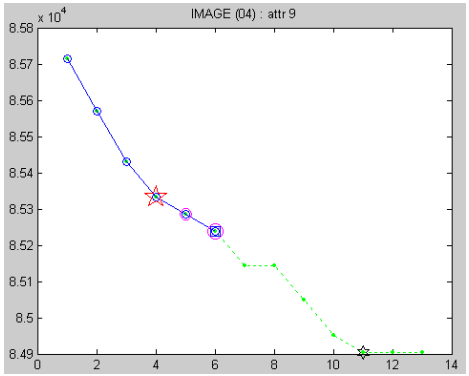
(f) ***NOI*** Searching Chain on Attr. No. 6



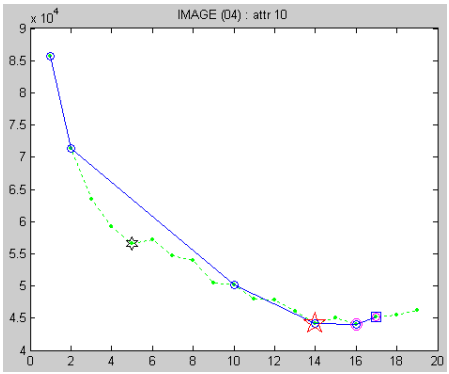
(g) ***NOI*** Searching Chain on Attr. No. 7



(h) ***NOI*** Searching Chain on Attr. No. 8

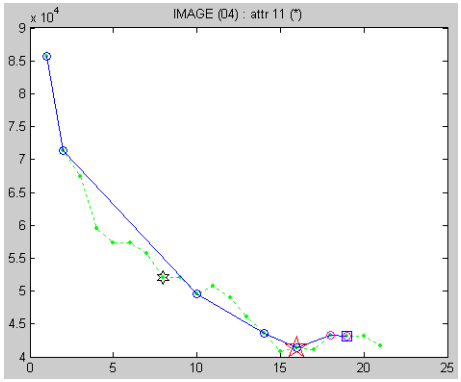


(i) ***NOI*** Searching Chain on Attr. No. 9

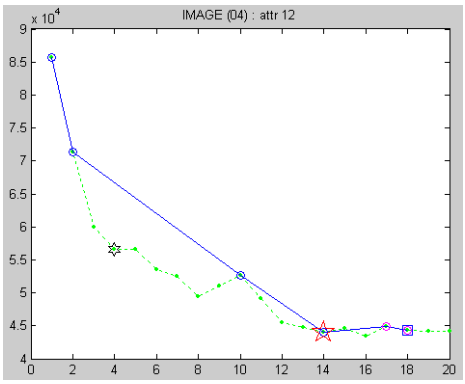


(j) ***NOI*** Searching Chain on Attr. No. 10

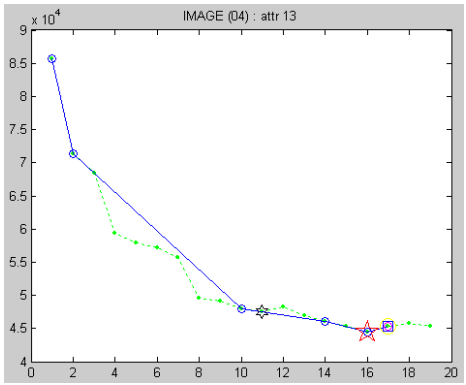
Figure 4.8 Searching Chain for *NOI* Values on *Image-Segmentation* at $\beta = 0.4$ (Continued)



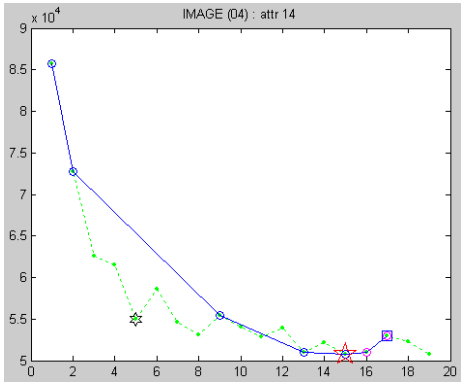
(k) ***NOI*** Searching Chain on Attr. No. 11



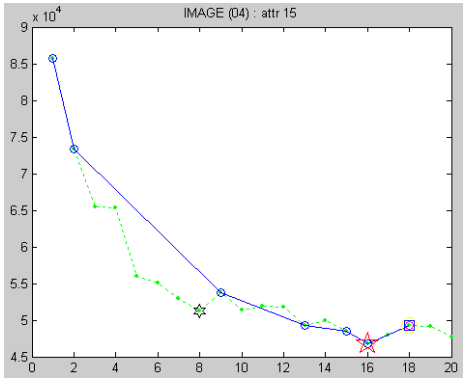
(l) ***NOI*** Searching Chain on Attr. No. 12



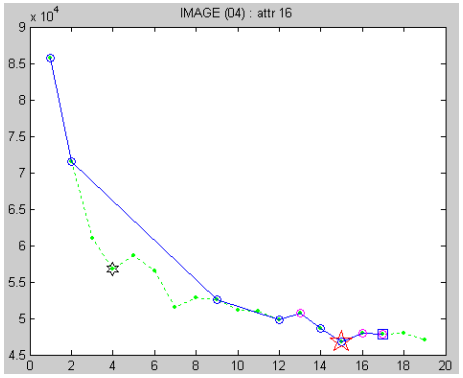
(m) ***NOI*** Searching Chain on Attr. No. 13



(n) ***NOI*** Searching Chain on Attr. No. 14

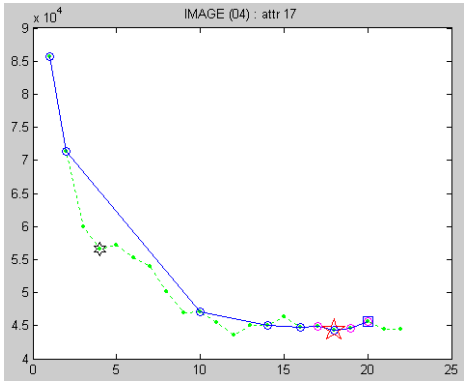


(o) ***NOI*** Searching Chain on Attr. No. 15

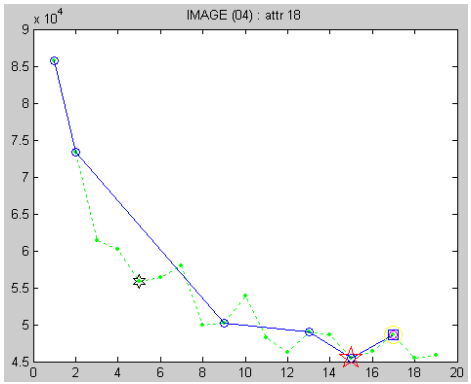


(p) ***NOI*** Searching Chain on Attr. No. 16

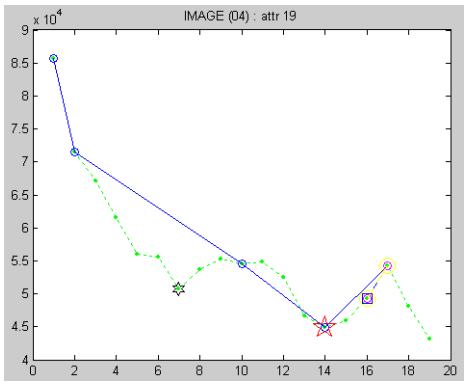
Figure 4.8 Searching Chain for *NOI* Values on *Image-Segmentation* at $\beta = 0.4$ (Continued)



(q) ***NOI*** Searching Chain on Attr. No. 17



(r) ***NOI*** Searching Chain on Attr. No. 18



(s) ***NOI*** Searching Chain on Attr. No. 19

Figure 4.8 Searching Chain for *NOI* Values on *Image-Segmentation* at $\beta = 0.4$ (Continued)

4.1.2 Experimental Results from Subspace Clustering Step

The Subspace Clustering procedure recursively partitions data space into subspaces using one attribute at a time for a partitioning dimension. The sequence of attributes used for the recursive partitioning follows the ascending order given by the Dimension Ordering step. Each partitioning step uses the same gradient descent approach as in the Dimension Ordering step to find the optimal noi value for the partitioning. After each of the partitioning, the residual $q(x)$ value continues decreasing or at least remains the same. The gradually improved $q(x)$ values during the recursive partitioning applied on the four datasets are shown in Table 4.5 and Table 4.6, at β value 0.1 and 0.4 respectively.

Table 4.5 $Q(x)$ Results Achieved during the Recursive Partitioning Step at $\beta = 0.1$

(a) Iris-Plants ($\beta = 0.1$)

Attribute No.	*Q(x)* Value	Attribute No.	*Q(x)* Value
4	0.04000	1	0.01333
3	0.01333	2	***0.00667***

(b) Pima-Indian Diabetes ($\beta = 0.1$)

Attribute No.	*Q(x)* Value	Attribute No.	*Q(x)* Value
2	0.25260	7	0.13843
8	0.24219	5	0.10338
6	0.23017	4	0.06727
1	0.19241	3	***0.04218***

(c) Vehicle Silhouettes ($\beta = 0.1$)

Attribute No.	*Q(x)* Value	Attribute No.	*Q(x)* Value
7	0.46432	4	0.03586
8	0.44932	10	0.02759
9	0.43204	18	0.02208
12	0.42849	17	0.02090
11	0.41313	13	0.01971
3	0.29256	14	0.01853
1	0.18064	15	0.01735
2	0.07270	16	0.01735
6	0.07225	5	***0.01735***

(d) Image-Segmentation ($\beta = 0.1$)

Attribute No.	*Q(x)* Value	Attribute No.	*Q(x)* Value
11	0.40742	14	0.03627
10	0.39690	8	0.03627
12	0.38785	6	0.03627
17	0.37933	1	0.01640
13	0.37933	4	0.01640
18	0.26975	5	0.01640
19	0.15246	7	0.01640
16	0.14187	9	0.01640
15	0.12664	3	***0.01640***
2	0.03960		

Table 4.6 $Q(x)$ Results Achieved during the Recursive Partitioning Step at $\beta = 0.4$

(a) Iris-Plants ($\beta = 0.4$)

Attribute No.	*Q(x)* Value	Attribute No.	*Q(x)* Value
4	0.04000	1	0.01333
3	0.01333	**2**	***0.00667***

(b) Pima-Indian Diabetes ($\beta = 0.4$)

Attribute No.	*Q(x)* Value	Attribute No.	*Q(x)* Value
2	0.25260	5	0.23307
6	0.25260	7	0.22266
1	0.25260	4	0.21615
8	0.24219	**3**	***0.19141***

Table 4.6 (Continued)

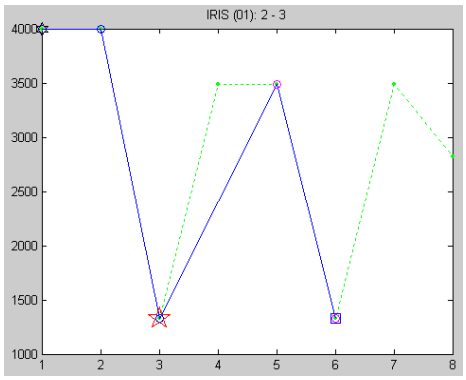
(c) Vehicle Silhouettes ($\beta = 0.4$)

Attribute No.	*Q(x)* Value	Attribute No.	*Q(x)* Value
7	0.48495	6	0.08704
8	0.47458	17	0.04803
9	0.45865	18	0.03385
12	0.45422	4	0.02912
11	0.44226	13	0.02321
3	0.33706	14	0.02203
1	0.23006	15	0.01848
10	0.11659	16	0.01730
2	0.08704	5	***0.01730***

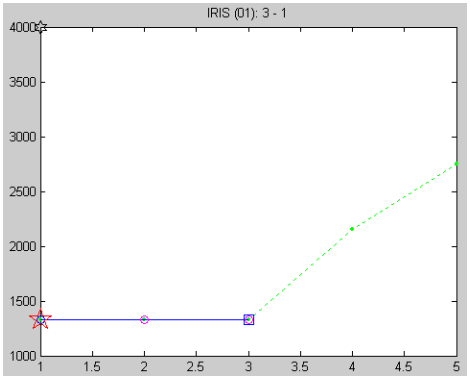
(d) Image-Segmentation ($\beta = 0.4$)

Attribute No.	*Q(x)* Value	Attribute No.	*Q(x)* Value
11	0.41397	14	0.05190
12	0.40048	8	0.05190
10	0.39714	6	0.05190
17	0.39286	1	0.03571
13	0.39286	4	0.03571
19	0.21968	5	0.03571
18	0.17333	7	0.03571
15	0.16285	9	0.03571
16	0.14131	3	***0.03571***
2	0.05952		

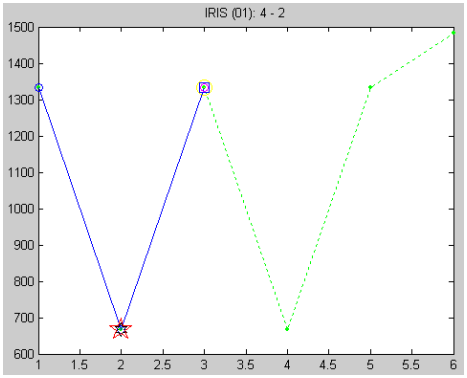
Figure 4.9 (a)-(c) show the searching chains for the optimal residual $q(x)$ during the second partitioning step till the last step for the *Iris-Plants* dataset, Figure 4.10 (a)-(g) for the *Pima-Indian Diabetes* dataset, Figure 4.11 (a)-(q) for the *Vehicle Silhouettes* dataset, and Figure 4.12 (a)-(r) for the *Image-Segmentation* dataset, all at $\beta = 0.1$. Notice that the first partitioning step has be performed during the Dimension Ordering step, so the chain is the same and therefore omitted. Figure 4.13 (a)-(c), Figure 4.14 (a)-(g), Figure 4.15 (a)-(q), and Figure 4.16 (a)-(r) demonstrates the same scenarios, but at $\beta = 0.4$. The meanings of the symbols used in the figures are the same as mentioned in section 4.1.1.



(a) ***NOI*** Search Chain on the 2nd partitioning



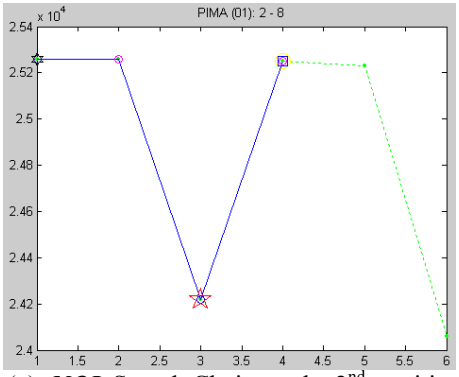
(b) ***NOI*** Search Chain on the 3rd partitioning



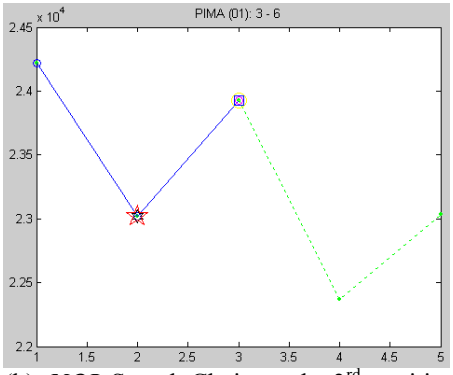
(c) ***NOI*** Search Chain on the 4th partitioning

Figure 4.9 Searching Chains for Subspace Partitioning on *Iris-Plants* at $\beta = 0.1$

.

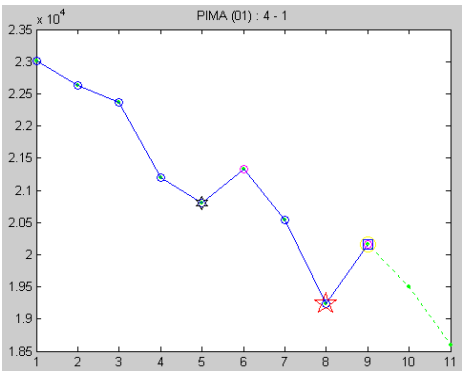


(a) ***NOI*** Search Chain on the 2nd partitioning

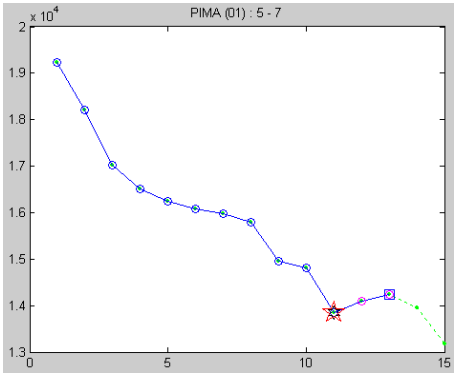


(b) ***NOI*** Search Chain on the 3rd partitioning

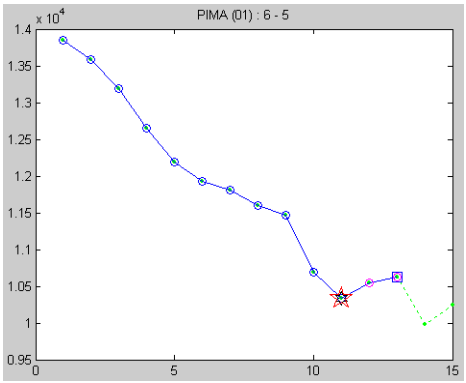
Figure 4.10 Searching Chains for Subspace Partitioning on *Pima-Indian Diabetes* at $\beta = 0.1$



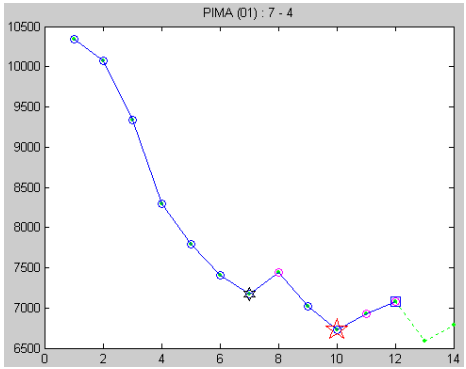
(c) ***NOI*** Search Chain on the 4th partitioning



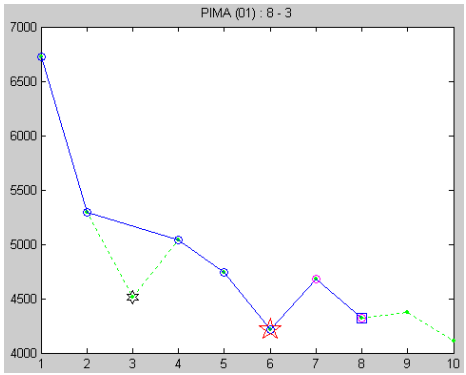
(d) ***NOI*** Search Chain on the 5th partitioning



(e) ***NOI*** Search Chain on the 6th partitioning

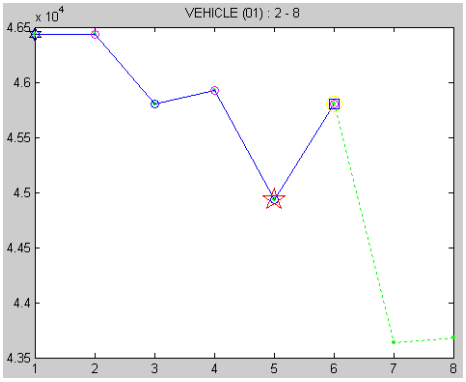


(f) ***NOI*** Search Chain on the 7th partitioning

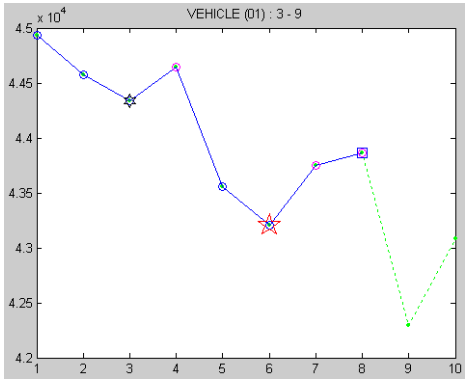


(g) ***NOI*** Search Chain on the 8th partitioning

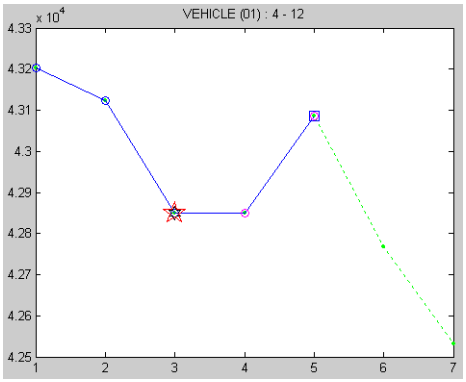
Figure 4.10 Searching Chains for Subspace Partitioning on *Pima-Indian Diabetes* at $\beta = 0.1$ (Continued)



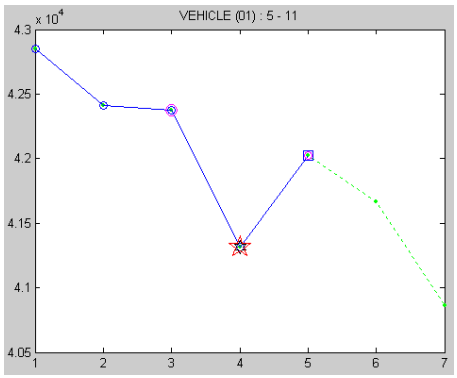
(a) ***NOI*** Search Chain on the 2nd partitioning



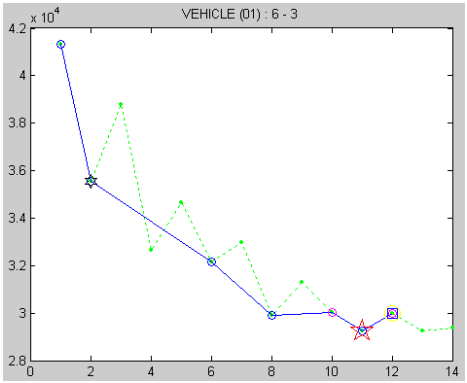
(b) ***NOI*** Search Chain on the 3rd partitioning



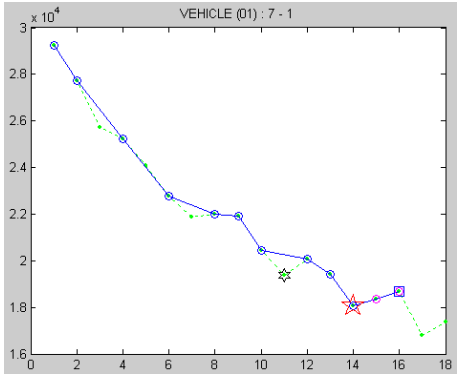
(c) ***NOI*** Search Chain on the 4th partitioning



(d) ***NOI*** Search Chain on the 5th partitioning

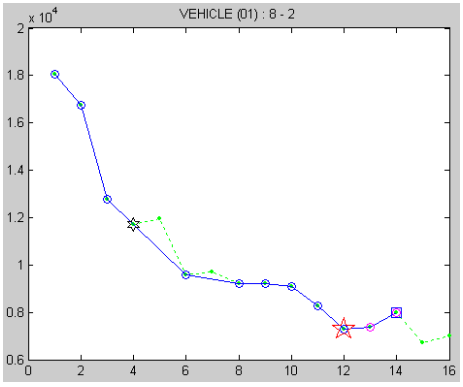


(e) ***NOI*** Search Chain on the 6th partitioning

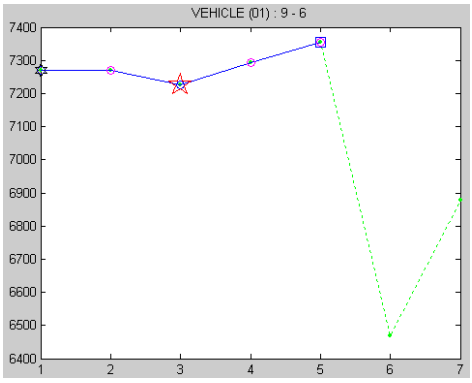


(f) ***NOI*** Search Chain on the 7th partitioning

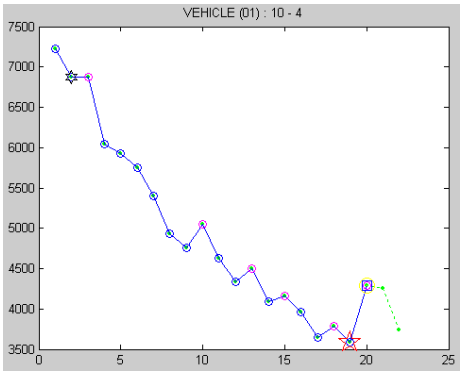
Figure 4.11 Searching Chain for Subspace Partitioning on *Vehicle Silhouettes* at $\beta = 0.1$



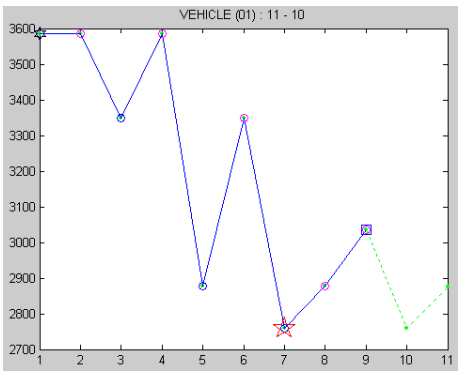
(g) ***NOI*** Search Chain on the 8th partitioning



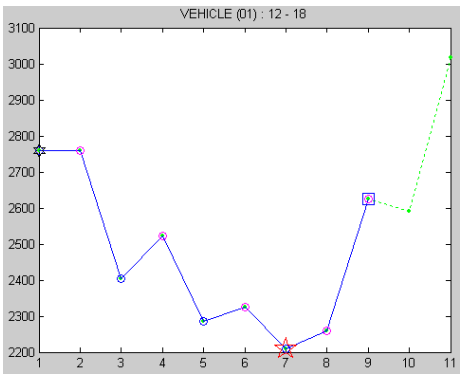
(h) ***NOI*** Search Chain on the 9th partitioning



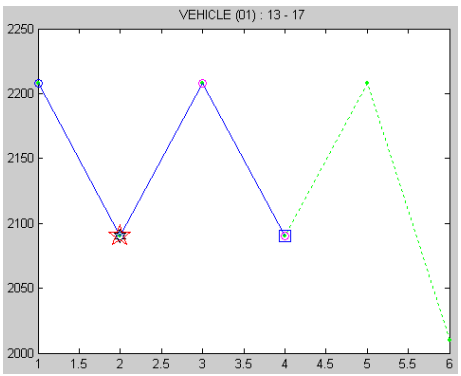
(i) ***NOI*** Search Chain on the 10th partitioning



(g) ***NOI*** Search Chain on the 11th partitioning

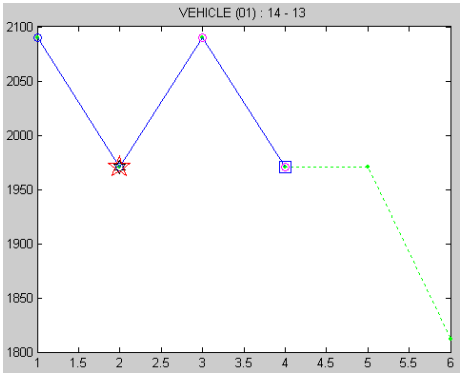


(k) ***NOI*** Search Chain on the 12th partitioning

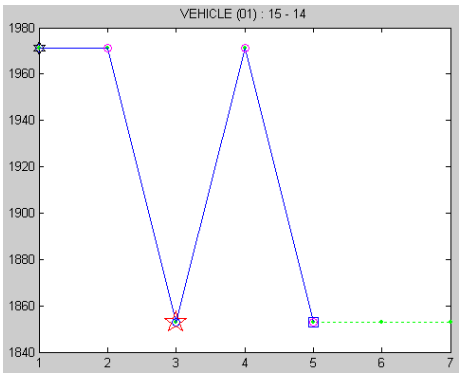


(l) ***NOI*** Search Chain on the 13th partitioning

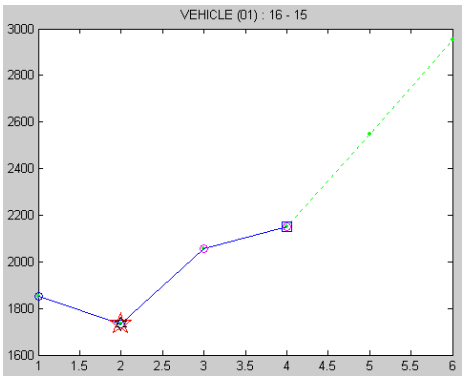
Figure 4.11 Searching Chain for Subspace Partitioning on *Vehicle Silhouettes* at $\beta = 0.1$ (Continued)



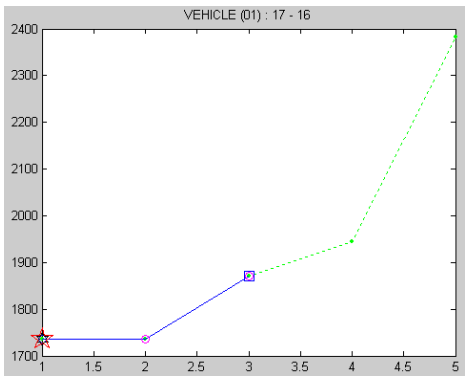
(m) ***NOI*** Search Chain on the 14th partitioning



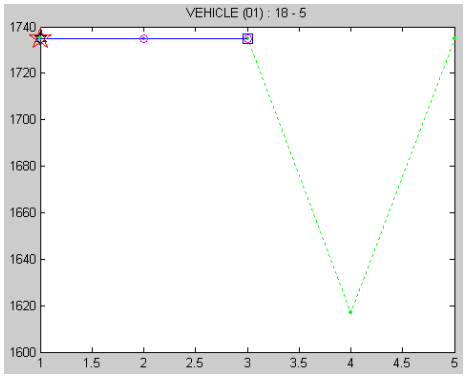
(n) ***NOI*** Search Chain on the 15th partitioning



(o) ***NOI*** Search Chain on the 16th partitioning

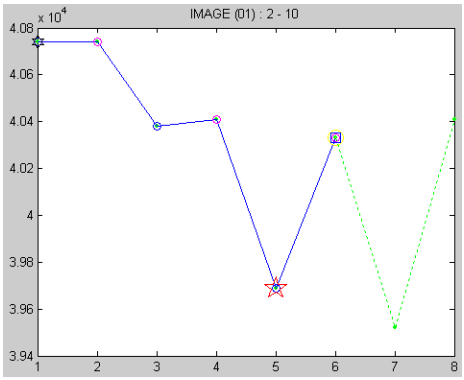


(p) ***NOI*** Search Chain on the 17th partitioning

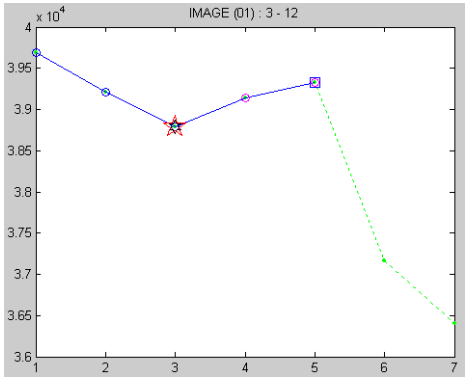


(q) ***NOI*** Search Chain on the 18th partitioning

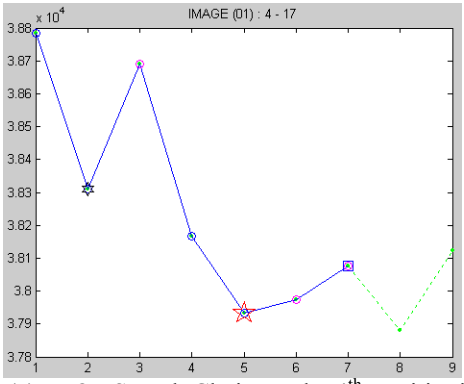
Figure 4.11 Searching Chain for Subspace Partitioning on *Vehicle Silhouettes* at $\beta = 0.1$ (Continued)



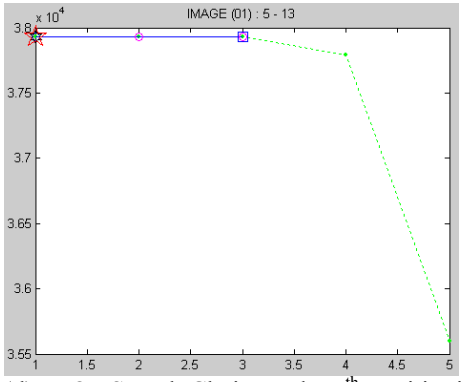
(a) ***NOI*** Searchg Chain on the 2nd partitioning



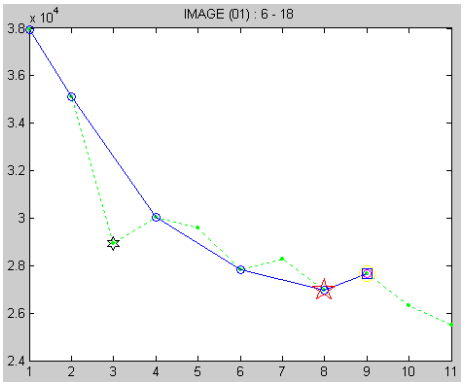
(b) ***NOI*** Search Chain on the 3rd partitioning



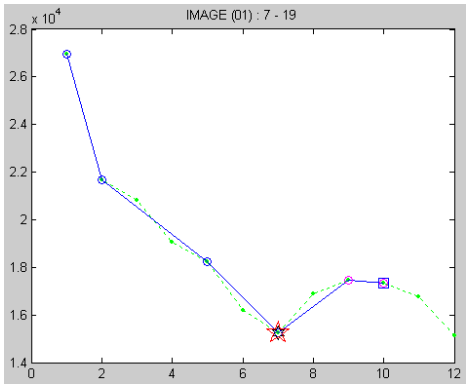
(c) ***NOI*** Search Chain on the 4th partitioning



(d) ***NOI*** Search Chain on the 5th partitioning

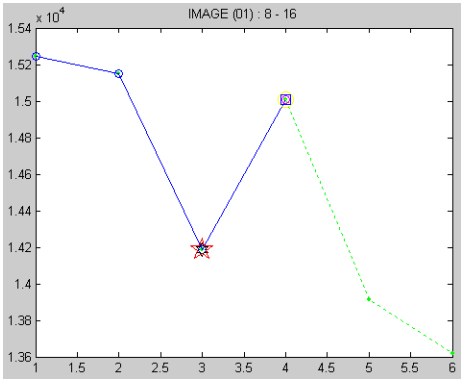


(e) ***NOI*** Search Chain on the 6th partitioning

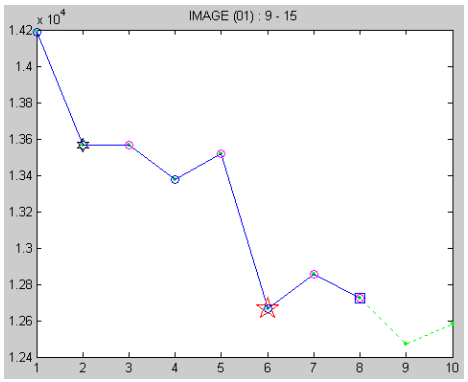


(f) ***NOI*** Search Chain on the 7th partitioning

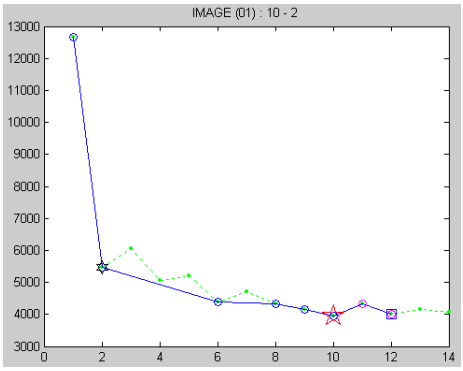
Figure 4.12 Searching Chains for Subspace Partitioning on *Image-Segmentation* at $\beta = 0.1$



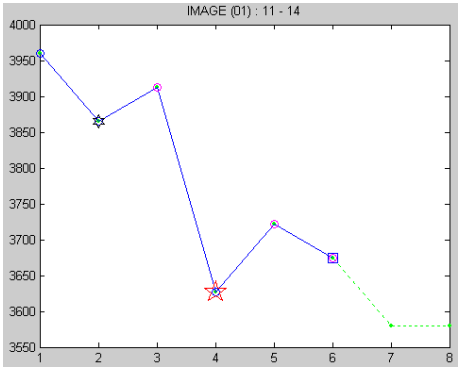
(g) ***NOI*** Search Chain on the 8th partitioning



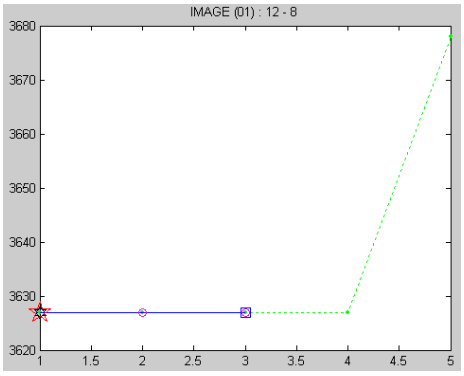
(h) ***NOI*** Search Chain on the 9th partitioning



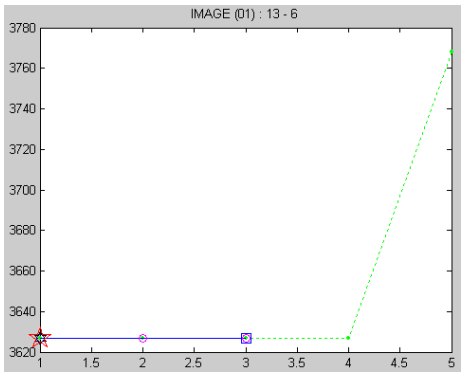
(i) ***NOI*** Search Chain on the 10th partitioning



(j) ***NOI*** Search Chain on the 11th partitioning

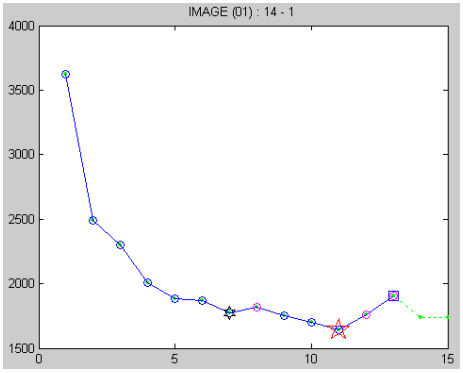


(k) ***NOI*** Search Chain on the 12th partitioning

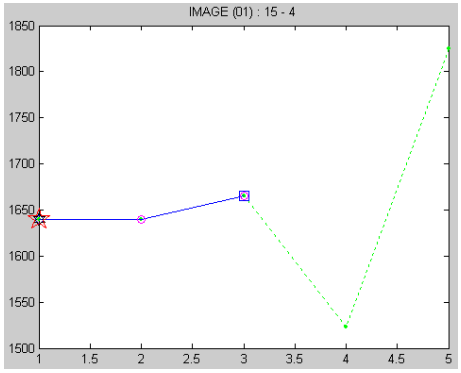


(l) ***NOI*** Search Chain on the 13th partitioning

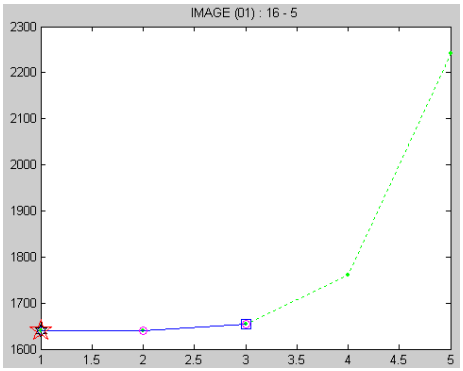
Figure 4.12 Searching Chains for Subspace Partitioning on *Image-Segmentation* at $\beta = 0.1$ (Continued)



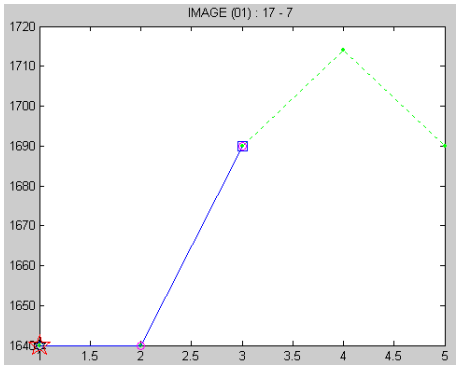
(m) ***NOI*** Search Chain on the 14th partitioning



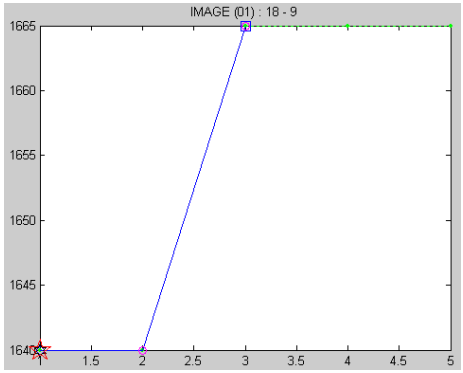
(n) ***NOI*** Search Chain on the 15th partitioning



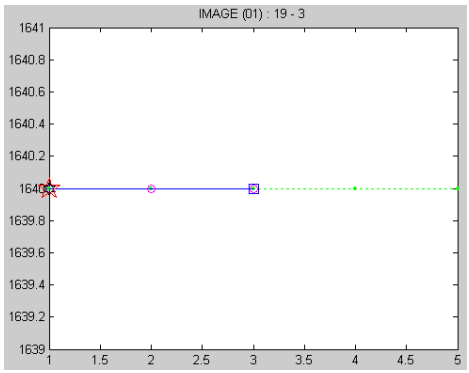
(o) ***NOI*** Search Chain on the 16th partitioning



(p) ***NOI*** Search Chain on the 17th partitioning

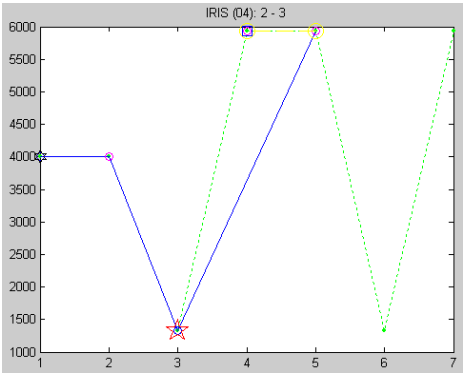


(q) ***NOI*** Search Chain on the 18th partitioning

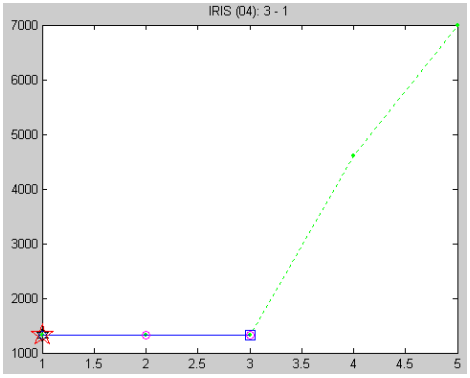


(r) ***NOI*** Search Chain on the 19th partitioning

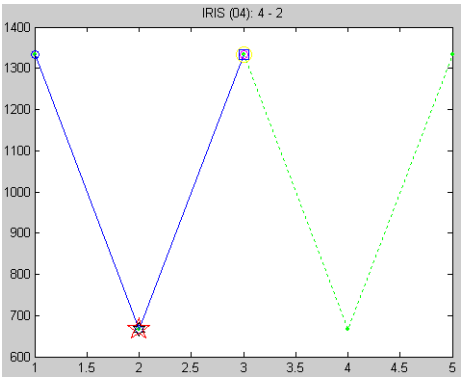
Figure 4.12 Searching Chains for Subspace Partitioning on *Image-Segmentation* at $\beta = 0.1$ (Continued)



(a) ***NOI*** Search Chain on the 2nd partitioning

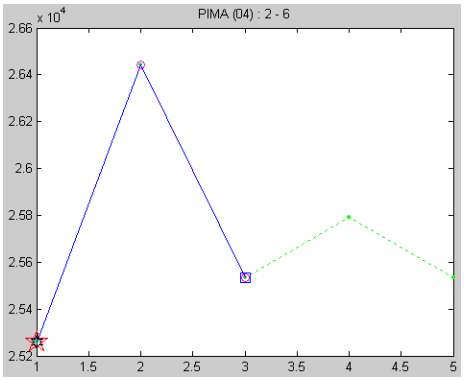


(b) ***NOI*** Search Chain on the 3rd partitioning

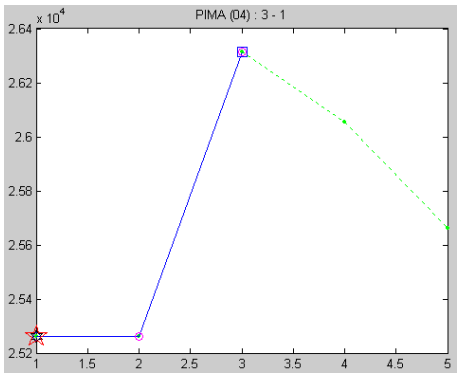


(c) ***NOI*** Search Chain on the 4th partitioning

Figure 4.13 Searching Chains for Subspace Partitioning on *Iris-plants* at $\beta = 0.4$

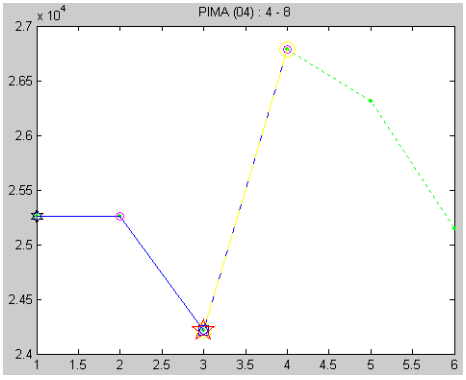


(a) ***NOI*** Search Chain on the 2nd partitioning

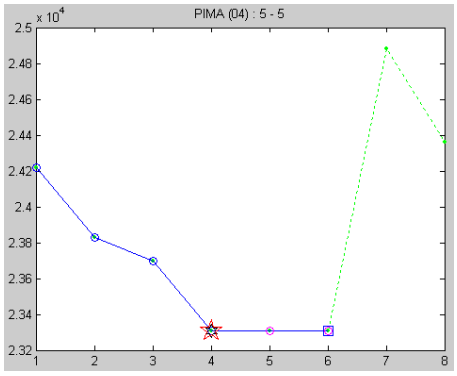


(b) ***NOI*** Search Chain on the 3rd partitioning

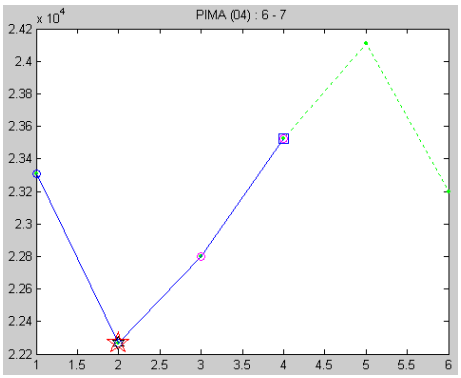
Figure 4.14 Searching Chains for Subspace Partitioning on *Pima-Indian Diabetes* at $\beta = 0.4$



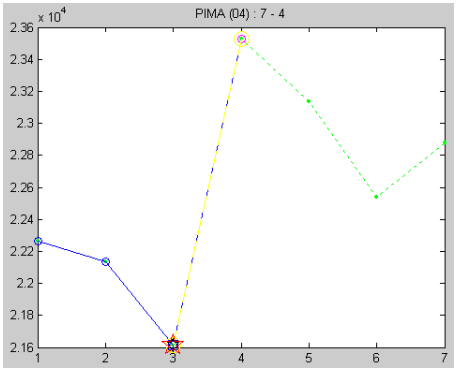
(c) ***NOI*** Search Chain on the 4th partitioning



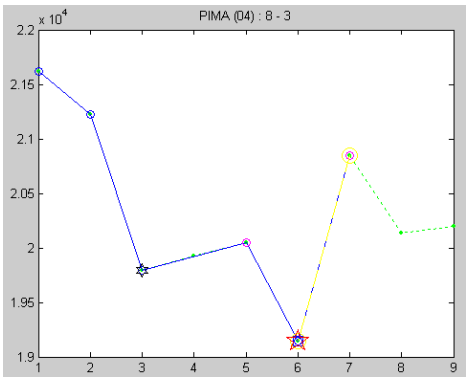
(d) ***NOI*** Search Chain on the 5th partitioning



(e) ***NOI*** Search Chain on the 6th partitioning

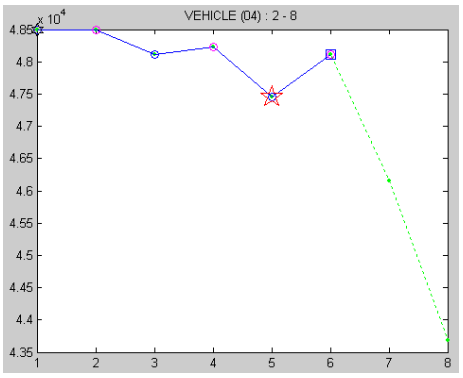


(f) ***NOI*** Search Chain on the 7th partitioning

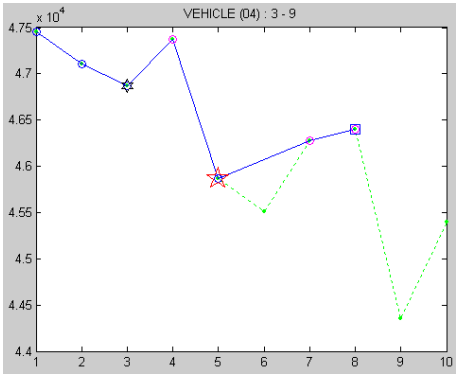


(g) ***NOI*** Search Chain on the 8th partitioning

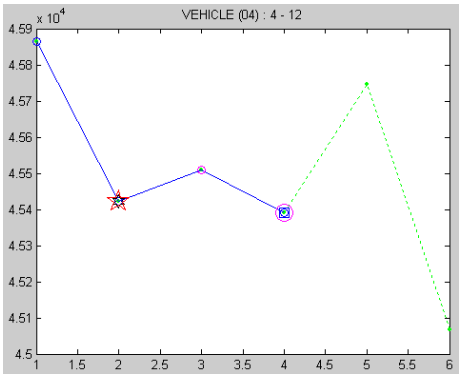
Figure 4.14 Searching Chains for Subspace Partitioning on *Pima-Indian Diabetes* at $\beta = 0.4$ (Continued)



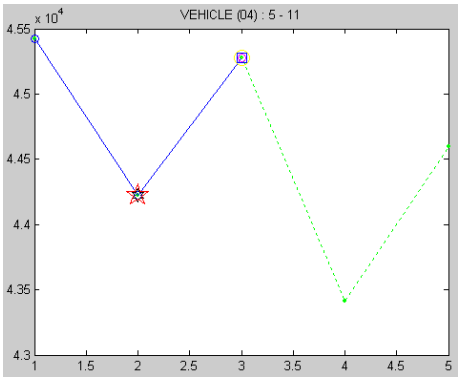
(a) ***NOI*** Search Chain on the 2nd partitioning



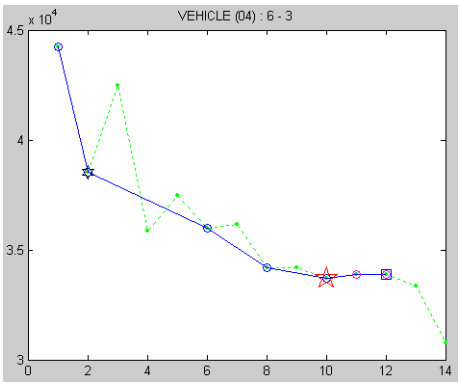
(b) ***NOI*** Search Chain on the 3rd partitioning



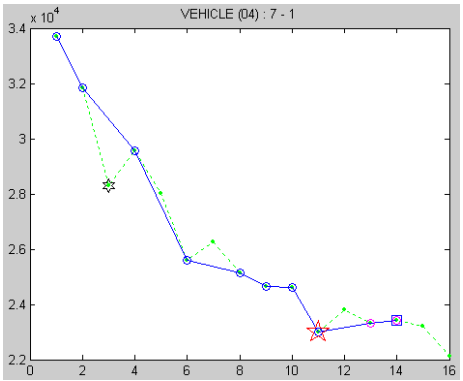
(c) ***NOI*** Search Chain on the 4th partitioning



(d) ***NOI*** Search Chain on the 5th partitioning

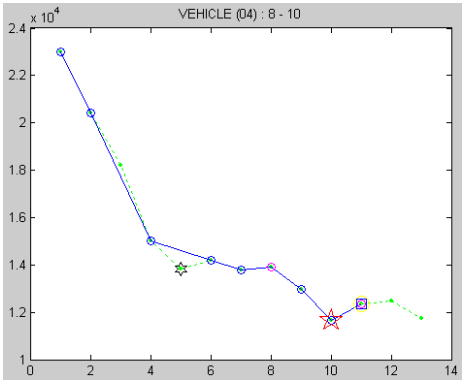


(e) ***NOI*** Search Chain on the 6th partitioning

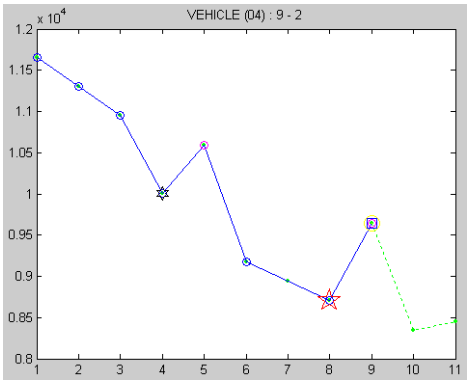


(f) ***NOI*** Search Chain on the 7th partitioning

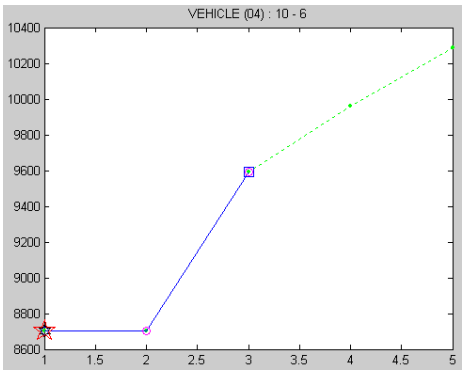
Figure 4.15 Searching Chain for Subspace Partitioning on *Vehicle Silhouettes* at $\beta = 0.4$



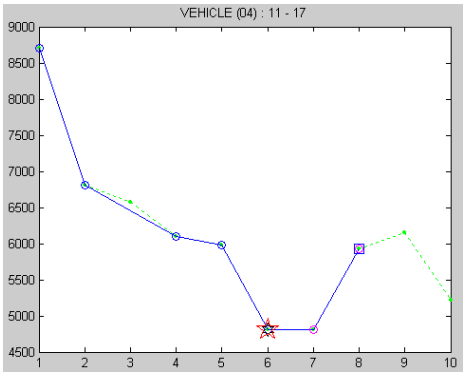
(g) ***NOI*** Search Chain on the 8th partitioning



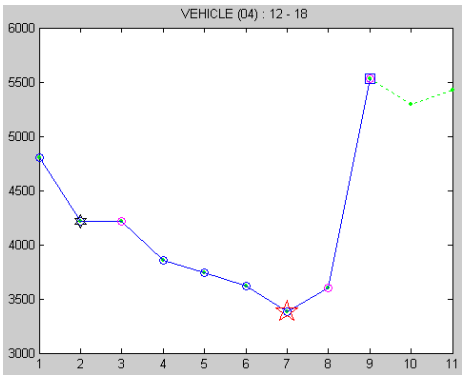
(h) ***NOI*** Search Chain on the 9th partitioning



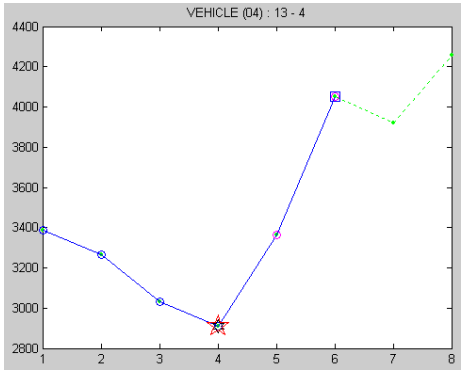
(i) ***NOI*** Search Chain on the 10th partitioning



(j) ***NOI*** Search Chain on the 11th partitioning

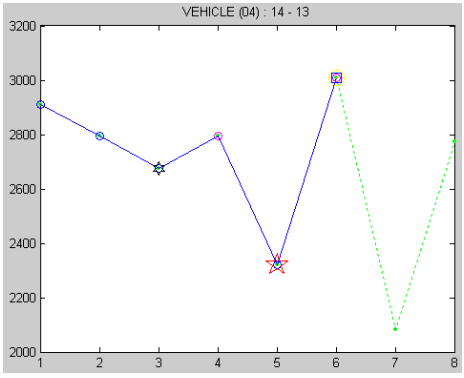


(k) ***NOI*** Search Chain on the 12th partitioning

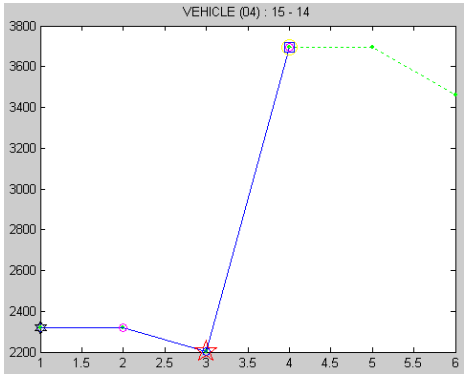


(l) ***NOI*** Search Chain on the 13th partitioning

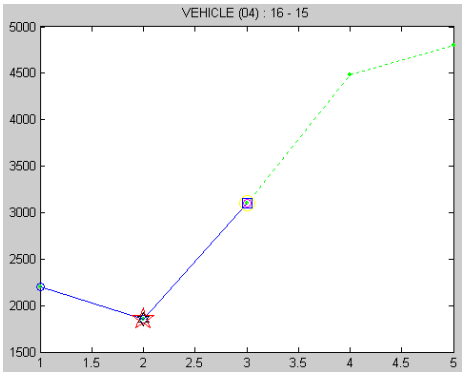
Figure 4.15 Searching Chain for Subspace Partitioning on *Vehicle Silhouettes* at $\beta = 0.4$ (Continued)



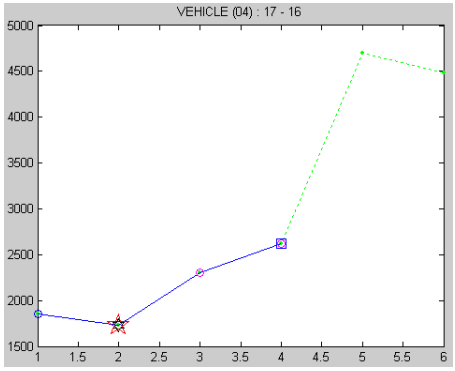
(m) ***NOI*** Search Chain on the 14th partitioning



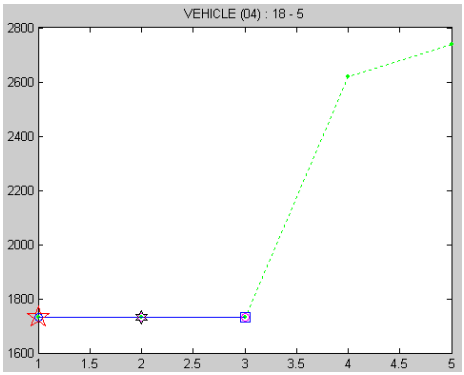
(n) ***NOI*** Search Chain on the 15th partitioning



(o) ***NOI*** Search Chain on the 16th partitioning

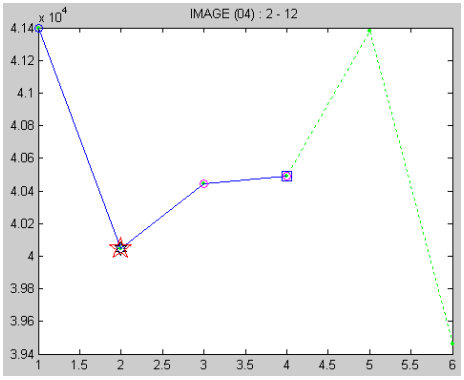


(p) ***NOI*** Search Chain on the 17th partitioning

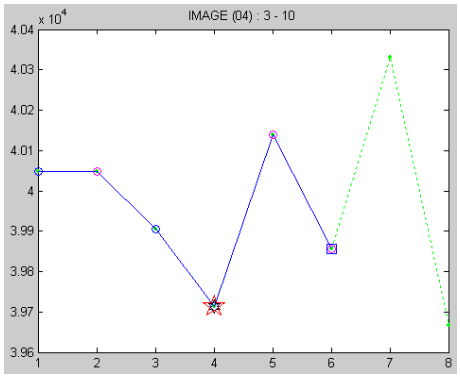


(q) ***NOI*** Search Chain on the 18th partitioning

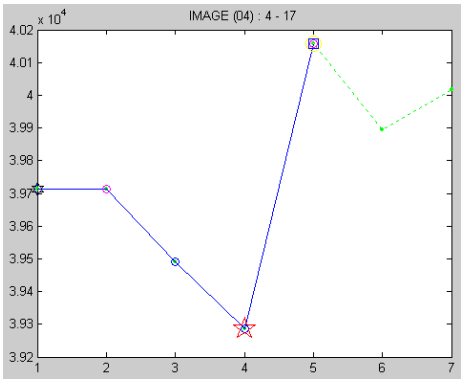
Figure 4.15 Searching Chain for Subspace Partitioning on *Vehicle Silhouettes* at $\beta = 0.4$ (Continued)



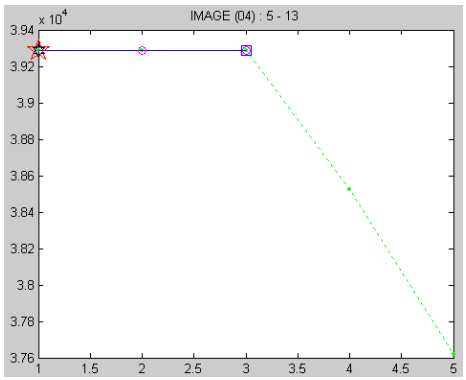
(a) ***NOI*** Search Chain on the 2nd partitioning



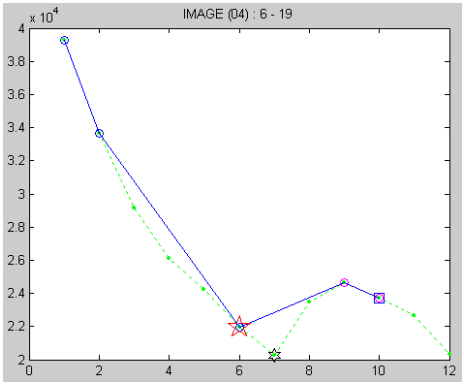
(b) ***NOI*** Search Chain on the 3rd partitioning



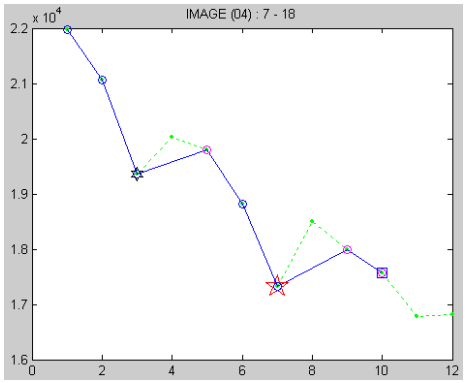
(c) ***NOI*** Search Chain on the 4th partitioning



(d) ***NOI*** Search Chain on the 5th partitioning

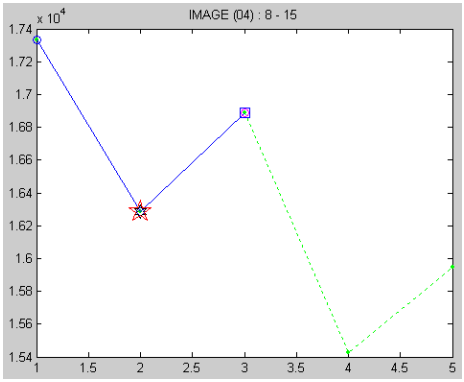


(e) ***NOI*** Search Chain on the 6th partitioning

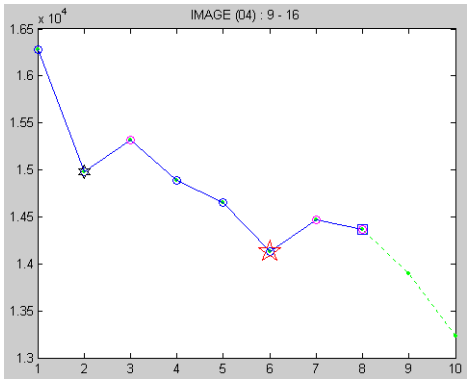


(f) ***NOI*** Search Chain on the 7th partitioning

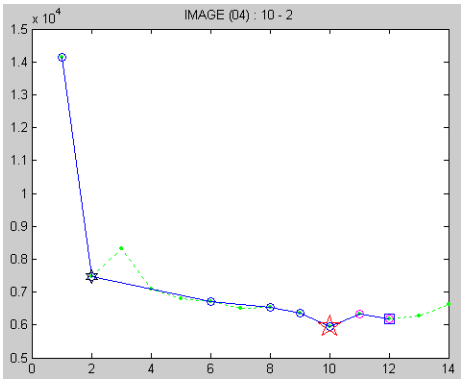
Figure 4.16 Searching Chains for Subspace Partitioning on *Image-Segmentation* at $\beta = 0.4$



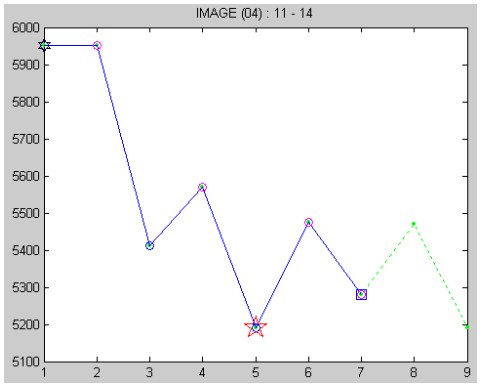
(g) ***NOI*** Search Chain on the 8th partitioning



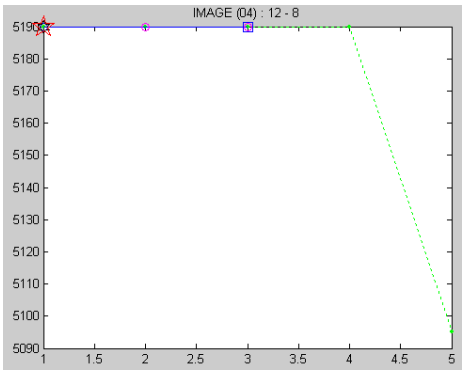
(h) ***NOI*** Search Chain on the 9th partitioning



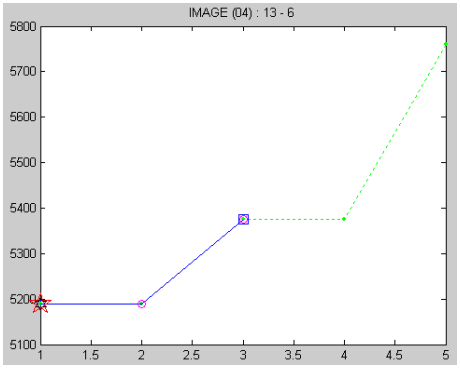
(i) ***NOI*** Search Chain on the 10th partitioning



(j) ***NOI*** Search Chain on the 11th partitioning

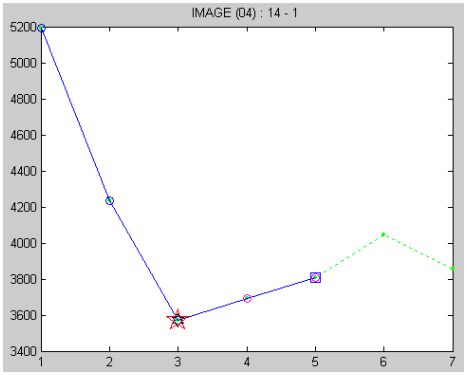


(k) ***NOI*** Search Chain on the 12th partitioning

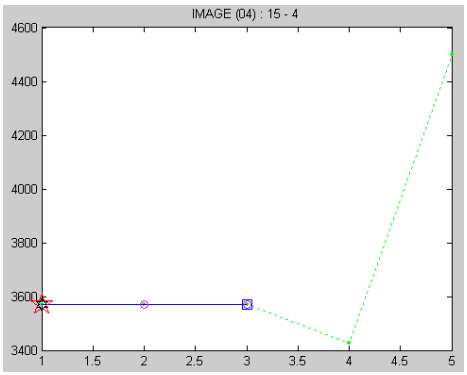


(l) ***NOI*** Search Chain on the 13th partitioning

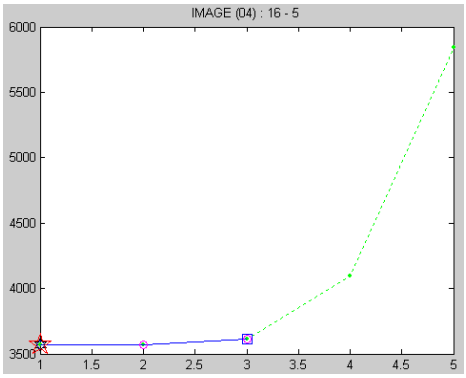
Figure 4.16 Searching Chains for Subspace Partitioning on *Image-Segmentation* at $\beta = 0.4$ (Continued)



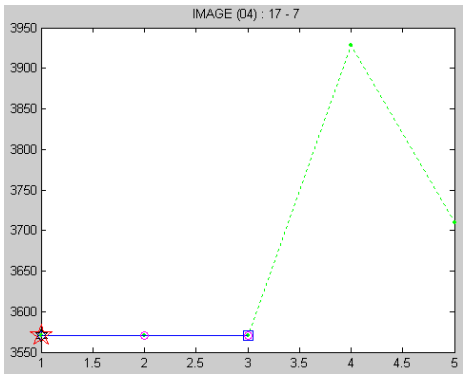
(m) ***NOI*** Search Chain on the 14th partitioning



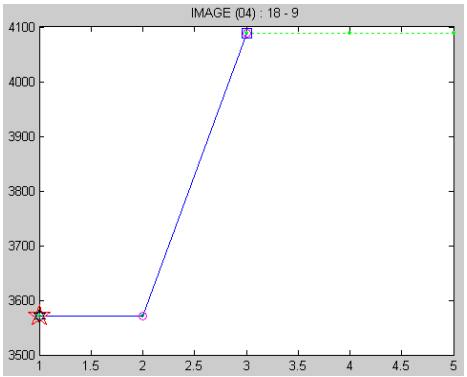
(n) ***NOI*** Search Chain on the 15th partitioning



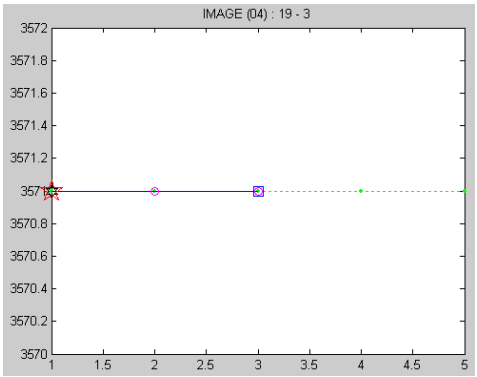
(o) ***NOI*** Search Chain on the 16th partitioning



(p) ***NOI*** Search Chain on the 17th partitioning



(q) ***NOI*** Search Chain on the 18th partitioning



(r) ***NOI*** Search Chain on the 19th partitioning

Figure 4.16 Searching Chains for Subspace Partitioning on *Image-Segmentation* at $\beta = 0.4$ (Continued)

4.1.3 Evaluation of the Clustering Performance

The clustering performance of the proposed algorithm on the four UCI datasets are finally evaluated in terms of the number of generated clusters, the clusters' purity (averaged ratio between the number of majority class data objects and the number of data objects in a cluster) , and the fitness function $q(x)$ value. The evaluation measurements are compared with those of the other five representative-based supervised clustering algorithms, which are Supervised Partitioning Around Medoids (SPAM), the Single Representative Insertion/Deletion Steepest Decent Hill Climbing with Randomized Start (SRIDHCR), and the Supervised Clustering using Evolutionary Computing (SCEC), all in Zeidat et al. (2006), the Supervised Growing Neural Gas (SGNG) and the Robust Supervised Growing Neural Gas (RSGNG) in Apirak Jirayusakul (2007). The performance comparisons on the *Iris-Plants*, the *Pima-Indian Diabetes*, the *Vehicle Silhouettes*, and the *Image-Segmentation* dataset are shown in Table 4.7, Table 4.8, Table 4.9, and Table 4.10 respectively, for 0.1 and 0.4 of β's values. Note that only the first three of the five algorithms are compared with the proposed algorithm for 0.4 of β's value since the three measurements are available on the three algorithms. In addition, the experiments on the proposed algorithm were also conducted with varying values of parameters, i.e. *maxnoi*, η and γ, to see how these parameters can affect the performances of the proposed algorithm.

Table 4.7 Performance Comparison on *Iris-Plants* Dataset

(a) Iris-Plants [on β = 0.1]

Algorithm	No. of Clusters	Cluster Purity	*Q(x)* Value
SCEC	5	0.993	0.018
SREDHCR	3	0.980	0.020
SPAM	3	0.973	0.027
SGNG	3	0.973	0.027
	5	0.986	0.026
RSGNG	3	0.973	0.027
	5	0.986	0.026
Proposed Algorithm			
(η = 0.5, maxnoi = 100, γ = 0.3)	***3***	***0.993***	***0.007***
(η = 0.4, maxnoi = 100, γ = 0.3)	*3*	*0.987*	*0.013*
(η = 0.6, maxnoi = 100, γ = 0.3)	*3*	*0.987*	*0.013*
(η = 0.5, maxnoi = 50, γ = 0.3)	*3*	*0.987*	*0.013*
(η = 0.5, maxnoi = 200, γ = 0.3)	*6*	*1.000*	*0.014*
(η = 0.5, maxnoi = 100, γ = 0.2)	*3*	*0.987*	*0.013*
(η = 0.5, maxnoi = 100, γ = 0.4)	*3*	*0.987*	*0.013*

(b) Iris-Plants [on β = 0.4]

Algorithm	No. of Clusters	Cluster Purity	*Q(x)* Value
SCEC	3	0.987	0.013
SREDHCR	3	0.987	0.013
SPAM	3	0.973	0.027
Proposed Algorithm			
(η = 0.5, maxnoi = 100, γ = 0.3)	***3***	***0.993***	***0.007***
(η = 0.4, maxnoi = 100, γ = 0.3)	*3*	*0.987*	*0.013*
(η = 0.6, maxnoi = 100, γ = 0.3)	*3*	*0.987*	*0.013*
(η = 0.5, maxnoi = 50, γ = 0.3)	*3*	*0.987*	*0.013*
(η = 0.5, maxnoi = 200, γ = 0.3)	*6*	*1.000*	*0.057*
(η = 0.5, maxnoi = 100, γ = 0.2)	*3*	*0.987*	*0.013*
(η = 0.5, maxnoi = 100, γ = 0.4)	*3*	*0.987*	*0.013*

As shown in Table 4.7, on the *Iris-Plants* dataset whose same class data objects are grouped in the well-defined oval shape clusters, the performance of the proposed algorithm is better than those of the other five algorithms both at $\beta = 0.1$ and $\beta = 0.4$, except at *maxnoi* value 200.

Table 4.8 Performance Comparison on *Pima-Indian Diabetes* Dataset

(a) Pima-Indian Diabetes [on β = 0.1]

Algorithm	No. of Clusters	Cluster Purity	*Q(x)* Value
SCEC	64	0.893	0.135
SREDHCR	45	0.859	0.164
SPAM	45	0.822	0.202
SGNG	45	0.880	0.144
	64	0.919	0.109
	75	0.941	0.090
RSGNG	45	0.863	0.161
	64	0.898	0.130
	75	0.911	0.120
Proposed Algorithm			
(η = 0.5, maxnoi = 100, γ = 0.3)	***37***	***0.979***	***0.042***
(η = 0.4, maxnoi = 100, γ = 0.3)	*30*	*0.974*	*0.045*
(η = 0.6, maxnoi = 100, γ = 0.3)	*60*	*0.990*	*0.038*
(η = 0.5, maxnoi = 50, γ = 0.3)	*20*	*0.939*	*0.077*
(η = 0.5, maxnoi = 200, γ = 0.3)	*63*	*0.982*	*0.046*
(η = 0.5, maxnoi = 100, γ = 0.2)	*25*	*0.961*	*0.056*
(η = 0.5, maxnoi = 100, γ = 0.4)	*20*	*0.961*	*0.054*

(b) Pima-Indian Diabetes [on β = 0.4]

Algorithm	No. of Clusters	Cluster Purity	*Q(x)* Value
SCEC	9	0.819	0.219
SREDHCR	2	0.776	0.224
SPAM	2	0.772	0.227
Proposed Algorithm			
(η = 0.5, maxnoi = 100, γ = 0.3)	***2***	***0.809***	***0.191***
(η = 0.4, maxnoi = 100, γ = 0.3)	*2*	*0.802*	*0.198*
(η = 0.6, maxnoi = 100, γ = 0.3)	*2*	*0.809*	*0.191*
(η = 0.5, maxnoi = 50, γ = 0.3)	*2*	*0.797*	*0.203*
(η = 0.5, maxnoi = 200, γ = 0.3)	*18*	*0.943*	*0.115*
(η = 0.5, maxnoi = 100, γ = 0.2)	*2*	*0.802*	*0.198*
(η = 0.5, maxnoi = 100, γ = 0.4)	*2*	*0.802*	*0.198*

On the *Pima-Indian Diabetes* dataset, shown in table 4.8, the proposed algorithm performs the best among the six algorithms in terms of both the $q(x)$ value and the cluster's purity when using β value 0.1. On β value of 0.4, its performance is rather comparable with the other three algorithms.

Table 4.9 shows that the proposed algorithm outperforms the other algorithms on the *Vehicle Silhouettes* dataset in terms of the *q(x)* value, the cluster purity, and the number of clusters generated, on both values of β.

Table 4.9 Performance Comparison on *Vehicle Silhouettes* Dataset

(a) Vehicle Silhouettes [on β = 0.1]

Algorithm	No. of Clusters	Cluster Purity	*Q(x)* Value
SCEC	132	0.923	0.116
SREDHCR	65	0.835	0.192
SPAM	65	0.764	0.263
SGNG	65	0.861	0.166
	109	0.920	0.115
	132	0.946	0.093
RSGNG	65	0.873	0.154
	109	0.937	0.098
	132	0.955	0.084
Proposed Algorithm			
(η = 0.5, maxnoi = 100, γ = 0.3)	***43***	***1.000***	***0.021***
(η = 0.4, maxnoi = 100, γ = 0.3)	*23*	*0.998*	*0.017*
(η = 0.6, maxnoi = 100, γ = 0.3)	*33*	*0.996*	*0.022*
(η = 0.5, maxnoi = 50, γ = 0.3)	*34*	*0.999*	*0.020*
(η = 0.5, maxnoi = 200, γ = 0.3)	*64*	*0.996*	*0.030*
(η = 0.5, maxnoi = 100, γ = 0.2)	*10*	*0.998*	*0.011*
(η = 0.5, maxnoi = 100, γ = 0.4)	*21*	*1.000*	*0.014*

(b) Vehicle Silhouettes [on β = 0.4]

Algorithm	No. of Clusters	Cluster Purity	*Q(x)* Value
SCEC	61	0.857	0.247
SREDHCR	56	0.835	0.265
SPAM	56	0.754	0.345
Proposed Algorithm			
(η = 0.5, maxnoi = 100, γ = 0.3)	***11***	***0.993***	***0.043***
(η = 0.4, maxnoi = 100, γ = 0.3)	*5*	*0.992*	*0.022*
(η = 0.6, maxnoi = 100, γ = 0.3)	*7*	*0.988*	*0.036*
(η = 0.5, maxnoi = 50, γ = 0.3)	*9*	*0.988*	*0.043*
(η = 0.5, maxnoi = 200, γ = 0.3)	*16*	*0.994*	*0.054*
(η = 0.5, maxnoi = 100, γ = 0.2)	*9*	*0.995*	*0.035*
(η = 0.5, maxnoi = 100, γ = 0.4)	*9*	*0.995*	*0.035*

Still in Table 4.10, the proposed algorithm gives better performance on the *image-Segmentation* dataset for both the *q(x)* values and the number of clusters on both β values.

Table 4.10 Performance Comparison on *Image-Segmentation* Dataset

(a) Image-Segmentation [on β = 0.1]

Algorithm	No. of Clusters	Cluster Purity	*Q(x)* Value
SCEC	60	0.989	0.026
SREDHCR	53	0.980	0.035
SPAM	53	0.944	0.071
SGNG	42	0.967	0.046
	53	0.971	0.044
	60	0.977	0.039
RSGNG	42	0.959	0.054
	53	0.963	0.052
	60	0.969	0.047
Proposed Algorithm			
(η = 0.5, maxnoi = 100, γ = 0.3)	***19***	***0.990***	***0.018***
(η = 0.4, maxnoi = 100, γ = 0.3)	*17*	*0.990*	*0.017*
(η = 0.6, maxnoi = 100, γ = 0.3)	*28*	*0.995*	*0.015*
(η = 0.5, maxnoi = 50, γ = 0.3)	*30*	*0.995*	*0.016*
(η = 0.5, maxnoi = 200, γ = 0.3)	*37*	*0.995*	*0.017*
(η = 0.5, maxnoi = 100, γ = 0.2)	*21*	*0.989*	*0.019*
(η = 0.5, maxnoi = 100, γ = 0.4)	*29*	*0.993*	*0.017*

(b) Image-Segmentation [on β = 0.4]

Algorithm	No. of Clusters	Cluster Purity	*Q(x)* Value
SCEC	28	0.969	0.069
SREDHCR	32	0.970	0.074
SPAM	32	0.940	0.103
Proposed Algorithm			
(η = 0.5, maxnoi = 100, γ = 0.3)	***14***	***0.984***	***0.039***
(η = 0.4, maxnoi = 100, γ = 0.3)	*13*	*0.980*	*0.041*
(η = 0.6, maxnoi = 100, γ = 0.3)	*15*	*0.990*	*0.034*
(η = 0.5, maxnoi = 50, γ = 0.3)	*13*	*0.985*	*0.037*
(η = 0.5, maxnoi = 200, γ = 0.3)	*25*	*0.986*	*0.051*
(η = 0.5, maxnoi = 100, γ = 0.2)	*14*	*0.988*	*0.035*
(η = 0.5, maxnoi = 100, γ = 0.4)	*13*	*0.984*	*0.038*

As a conclusion, the results reported in Table 4.7 thru Table 4.10 do show that the proposed algorithm yields the best solutions among the six algorithms in both values of β. Furthermore, the numbers of clusters generated by the other five

algorithms in the last three datasets are remarkably higher than those by the proposed algorithm. These may due to the natures of representative-based clustering algorithms of the other five algorithms that incline to create a large number of global-shape clusters. The parameter value of *maxnoi* has a few effects on the performances of the proposed algorithm if the value is too large. The *maxnoi* acts as a normalization factor for the *noi* value. So, when the *maxnoi* value becomes too large, the *noi* value as well as the gradient of the *noi* value become too small, and therefore can reduce the effectiveness of the gradient search. Also, it is possible that different dimensions can have different *maxnoi* values.

4.2 The Experiments on Synthetic Datasets

The experiments in this section are intended to affirm the effectiveness of the proposed algorithm under various proclaimed situations through *2-D* geometrical presentations. The algorithm was performed on four 2-D synthetic datasets from Apirak Jirayusakul (2007) representing 4 different scenarios: *Test-1*, *Test-2*, *Test-3*, and *Test-4*. The properties of the datasets are shown in Table 4.11. The parameters used in the experiments are as follows:

$$\beta = 0.1\,,\ \eta = 0.5\,,\ maxnoi = 100\,,\ \Upsilon = 0.3$$

The results of the experiments are graphically displayed in Figure 4.17 thru 4.20 where objects claimed as impurities are encircled with the dark color.

Table 4.11 Properties of the Synthetic Datasets Used in the Experiments

Dataset Name	No. of Examples	No. of Attributes	No. of Classes
Test-1	1,250	2	2
Test-2	1,000	2	6
Test-3	4,185	2	4
Test-4	3,421	2	4

Figure 4.17 illustrates the result from the experiment on *Test-1* dataset. It shows two pure cross-board shape clusters: *A* and *B*, one cluster per one individual class. This result confirms that the proposed algorithm has ability to identify any

irregular shape clusters. The result is quite different from those from Apirak Jirayusakul (2007: 61-62), the optimal value of which renders 25 prototypes, due to the representative-based nature which inclines to generate global shape clusters.

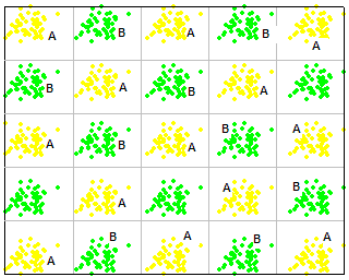


Figure 4.17 Result on *Test-1* Dataset

The result on *Test-2* dataset is shown in Figure 4.18. The proposed algorithm depicts 12 sparse various shape and density clusters: *A1, A2, B1, B2, C1, C2, C3*, *D1, D2, E1, E2* and *F,* with sparse-and-scattered impurity objects. There are two clusters which contain only single data object: *A3* and *E3*, and hence may be counted as outliers. The result is analogous to the optimal result generated by Apirak Jirayusakul (2007: 66), with the trivial exception that the clusters *A3* and *E3* are included into their major clusters.

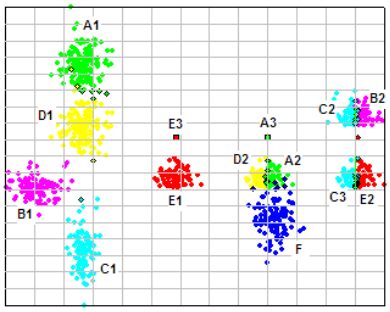


Figure 4.18 Result on *Test-2* Dataset.

The set of clusters shown in Figure 4.19 is the result from running the proposed algorithm on *Test-3* dataset. Fourteen crowded similar shape, size, and density clusters are delineated: *A1, A2, A3, A4, B1, B2, B3, B4, C1, C2, C3*, *C4, D1,*

and *D2*, most of which are overlapped, as can be seen surrounded by considerable number of impurity objects. The result is comparable to the optimal result from Apirak Jirayusakul (2007: 69-70), except that the number of clusters is higher due to the separation of the clusters *C3* into 2 adjacent clusters, according to its representative-based nature.

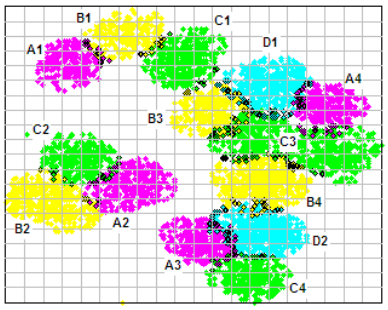


Figure 4.19 Result on *Test-3* Dataset.

The results from the experiment performed on *Test-4* dataset is shown in Figure 4.20. The proposed algorithm can identify 17 various size and density clusters: *A1, A2, A3, A4, A5, B1, B2, B3, B4, B5, C1, C2, C3, C4, C5, D1,* and *D2*. Some contain a small number of impurity objects in various locations inside the clusters. In Apirak Jirayusakul (2007: 73-74), the separation of both the cluster *A2* and *B3* into 2 adjacent clusters does confirm its representative-based concept.

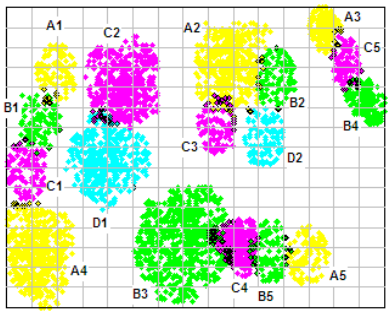


Figure 4.20 Result on *Test-4* Dataset.

As the conclusion, the results from the four experiments hence endorse the ability of the proposed algorithm in identifying clusters of any shapes and sizes without presuming any canonical form of data distribution.

CHAPTER 5

CONCLUSION AND FUTURE WORK

In this dissertation, a new approach in clustering data objects in the supervised manner is proposed. The proposed algorithm blends together the advantages of the grid-based clustering method and bottom-up subspace clustering method. This enables the algorithm to alleviate some of the drawbacks that exist in most of the recent clustering algorithms. That is, the proposed algorithm has ability to identify clusters of any shapes and sizes without presuming any canonical form for data distribution, needs no pre-specified number of clusters, and is insensitive to the order of the input data objects.

The proposed algorithm begins by gradually partitioning data space into equal-size non-empty grid cells using one dimension at a time. The greedy method is used to determine the order of dimensions that would give the best quality of clustering from the gradual partitioning, while, the gradient descent method is used to find the optimal number of intervals to be used for each partitioning. After all dimensions have been partitioned, any connected grid cells containing majority of data from the same class are merged into a cluster. By using the greedy and gradient descent methods in performing grid cell partitioning, the proposed algorithm can produce high quality clusters while reduce time to find the best partitioning and avoid the memory confinement problem during the process.

The results from the experiments do confirm that the proposed algorithm can cope with datasets of any shapes and sizes. On two-dimensional synthetic datasets, the proposed algorithm can identify clusters with different shapes and sizes correctly. On UCI datasets, the proposed algorithm also outperforms all other five supervised clustering algorithms, with smaller numbers of created clusters and lower degrees of impurity, especially on irregular-shape datasets.

A method to manage outliers is not mentioned in the proposed algorithm. Nevertheless, the algorithm can simply identify outliers by checking for clusters that

contain data objects fewer than the pre-defined value and reporting the data objects inside the clusters as outliers. A more sophisticated method can be developed based on some statistical characteristics of grid cells which might contain only outliers. For instance, during the Grid Cell Creation task, the mean and standard deviation values of the numbers of data objects of all grid cells can be computed. The grid cells with numbers of data objects significantly fewer than most other grid cells , e.g. less than the mean by more than four times of the standard deviation, can be considered as the grid cells with too few number of data objects and can be considered to contain only outliers. Therefore, they can then be deprived from further partitioning.

BIBLIOGRAPHY

Aggarwal, C.C.; Gates, S.C. and Yu, P.S. 1999. On the Merits of Building Categorization Systems by Supervised Clustering. In **Proceedings of the fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining KDD 99**. San Diego, CA: ACM. Pp. 352-356.

Agrawal, R.; Gehrke, J.; Gunopulos, D. and Raghavan, P. 1998. Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. In **Proceedings of ACM-SIGMOD International Conference on Management of Data (SIGMOD).** Seattle, WA: ACM. Pp. 94-105.

Aguilar, J.S.; Ruiz, R.; Riquelme, J.C. and Giráldez, R. 2001. SNN: A Supervised Clustering Algorithm. In **Proceedings of the 14th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE 2001).** Budapest: Springer-Verlag. Pp. 207-216.

Al-Harbi, S.H. and Rayward-Smith, V.J. 2006. Adapting K-means for Supervised Clustering. **Applied Intelligence.** 24(June): 219-226.

Apirak Jirayusakul. 2007. **Supervised Growing Neural Gas Algorithm in Clustering Analysis**. Doctoral dissertation, National Institute of Development Administration.

Apirak Jirayusakul and Surapong Auwatanamongkol. 2007. **A Supervised Growing Neural Gas Algorithm for Cluster Analysis**. International Journal of Hybrid Intelligent Systems. 4(December): 217-229.

Dettling, M. 2003. Revealing Predictive Gene Groups with Supervised Algorithms. In **Proceedings of the Conference in Distributed Statistical Computing, DSC 2003**. Vienna: DSC. Pp. 1-8.

Eick, C.F.; Zeidat, N. and Zhenghong, Z. 2004. Supervised Clustering - Algorithms and Benefits. In **Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI04).** Boca Raton, FL: IEEE. Pp. 774-776.

Finley, T. and Joachims, T. 2005. Supervised Clustering with Support Vector Machines. In **Proceedings of the International Conference on Machine Learning (ICML).** Bonn: ICML. Pp. 217-224.

Hinneburg, A. and Keim, D.A. 1998. An Efficient Approach to Clustering in Large Multimedia Databases with Noise. Retrieved April 15, 2008 from http://www.cs.ecu.edu/~dingq/CSCI6905/readings/kdd98-DENCLUE.pdf.

Hornik, K.; Leisch, F., and Zeileis, A. Eds. 2003. In **Proceedings of the 3rd International Workshop on Distributed Statistical Computing (DSC 2003)**. Vienna: DSC.

Kotsiantis, S.B. and Pintelas, P.E. 2004. Recent Advances in Clustering: A Brief Survey. **WSEAS Transactions on Information Science and Applications.** 1(1): 73-81.

Kunttu, I.; Lepisto, L.; Rauhamaa, J. and Visa A. n.d. Grid-based Clustering in the Content-based Organization of Large Image Databases. Retrieved April 15, 2008 from http://www.cs.tut.fi/~avisa/digger/Publications/WIAMIS2004_Kunttu.pdf.

Li, X. and Ye ,N. 2002. Grid-and Dummy-Cluster-Based Learning of Normal and Intrusive Clusters of Computer Intrusion Detection. **Journal of Quality and Reliability Engineering International**. 18(3): 231-242.

Li, X. and Ye, N. 2005. A Supervised Clustering Algorithm for Computer Intrusion Detection. **Knowledge and Information Systems.** 8(4): 498–509.

Li, X. and Ye ,N. 2006. A Supervised Clustering and Classification Algorithm for Mining Data with Mixed Variables. **IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans.** 36(March): 396-406.

Liao, W.K.; Liu, Y. and Choudhary, A. 2004. A Grid-based Clustering Algorithm Using Adaptive Mesh Refinement. In **the 7th Workshop on Mining Scientific and Engineering Data Sets 2004.** Held in conjunction with the 4th SIAM International Conference on Data Mining at Lake Buena Vista, Florida, April 24, 2004.

Park, N.H. and Lee, W.S. 2004. Statistical Grid-based Clustering over Data Streams. **SIGMODD Record**. 33(March): 32-37.

Parsans, L.; Haque, E. and Liu, H. 2004. Subspace Clustering for High Dimensional Data: A Review. **ACM SIGKDD Explorations**. 6(1): 90–105.

Pornpimol Bungkomkhun and Surapong Auwatanamongkol. 2009. Grid-based Supervised Clustering. In **Proceedings of December 2009 International Conference on Applied Mathematics and Computer Science, World Academy of Science, Engineering and Technology.** Bangkok: WASET. Pp. 536-543.

Procopiuc, C.M. 1997. Applications of Clustering Problems. Retrieved May 30, 2009 from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.39.2592&rep=rep1&type=pdf.

Sheikholeslami, G.; Chatterjee, S. and Zhang, A. 1998. WaveCluster: A Multi-Resolution Clustering Approach for Very Large Spatial Databases. In **Proceedings of the 24th International Conference on Very Large Databases (VLDB).** New York: VLDB. Pp. 428-439.

Sinkkonen, J.; Kaski, S. and Nikkila, J. 2002. Discriminative Clustering: Optimal Contingency Tables by Learning Metrics. In **Proceedings of the 13th European Conference on Machine Learning (ECML).** Springer-Verlag: ECLM. Pp. 418-430.

Slonim, N. and Tishby, N. 1999. Agglomerative Information Bottleneck. In **Proceedings of the 13rd Neural Information Processing Systems (NIPS)**. Pp. 617-623.

Tan, P-N.; Steinbach, M. and Kumar, V. 2006. **Introduction to Data Mining**. Hudson, NY: Hamilton.

Tishby, N.; Pereira, F.C. and Bialek, W. 1999. The Information Bottleneck Method. In **Proceedings of the 37th Annual Allerton Conference on Communication, Control, and Computing**. Urbana, IL. Pp. 368-377.

Qu, Y. and Xu, Z. 2004. Supervised Clustering Analysis for Microarray Data Based on Multivariate Gaussian Mixture. **Bioinformatics**. 20(12): 1905-1913.

University of California at Irving. 2008. Machine Learning Repository. Retrieved October 10, 2008 from http://www.ics.uci.edu/~mlearn/MLRepository.html.

Wang, W.; Yang, J. and Muntz, R. 1997. STING : A Statistical Information Grid Approach to Spatial Data Mining. Retrieved April 15, 2009 from http://fmdb.cs.ucla.edu/Treports/970006.pdf.

Ye, N. and Li, X. 2001. A Scalable Clustering Technique for Intrusion Signature Recognition. In **Proceedings of the 2nd IEEE Systems, Man, and Cybernetics Information Assurance Workshop.** West Point, NY: IEEE. Pp. 1-4.

Ye, N. and Li, X. 2005. Method for Classifying Data Using Clustering and Classification Algorithm Supervised. Retrieved April 15, 2009 from http://www.patentstorm.us/patents/6907436/fulltext.html.

Zeidat, N.; Eick, C.F. and Zhao, Z. 2006. Supervised Clustering: Algorithms and Application. Retrieved March 21, 2008 from http://www.cs.uh.edu/docs/cosc/technical-reports/2006/06_10.pdf.

BIOGRAPHY

NAME	Pornpimol Bungkomkhun
ACADEMIC BACKGROUND	B.S.(Statistics), Chulalongkorn University, 1979
	M.S.(Business Computer Information System), North Texas State University, 1984
PRESENT POSITION	Associate Director, Leam Chabang Engineering Technological College
EXPERIENCES	Programmer, Data Processing Division, Metropolitan Electricity Authority
	Head of Software Maintenance Department, Metropolitan Electricity Authority
	Special Instructor, Statistic Department, Faculty of Commerce and Accountancy, Chulalongkorn University
	Visitor Lecturer, Thai-Nichi Institute of Technology
	Head of Business Computer Department, South-East Asia Universi

Head of Computer Science Department, South-East Asia University

Associate Director, Leam Chabang Engineering Technological College

Conference: ‘Grid-based Supervised Clustering’, in **International Conference on Applied Mathematics and Computer Science, World Academy of Science, Engineering and Technology (WASET),** December 2009

Publication: 'Grid-based Supervised Clustering Algorithm Using Greedy and Gradient Descent Methods to Build Clusters', in **International Journal of Computer Science Issues (IJCSI)** Volume 9, Issue 3, May 2012