2015

Efficient Variations of the Quality Threshold Clustering Algorithm

Frank Loforte Jr.
Nova Southeastern University, floforte@me.com

Efficient Variations of the Quality Threshold Clustering Algorithm

by

Frank Loforte, Jr.

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in
Computer Information Systems

Graduate School of Computer and Information Sciences
Nova Southeastern University
2015

We hereby certify that this dissertation, submitted by Frank Loforte, conforms to acceptable standards and is fully adequate in scope and quality to fulfill the dissertation requirements for the degree of Doctor of Philosophy.

_____ _____

Sumitra Mukherjee, Ph.D. Date
Chairperson of Dissertation Committee

_____ _____

Francisco J. Mitropoulos, Ph.D. Date
Dissertation Committee Member

_____ _____

Michael J.Laszlo, Ph.D. Date
Dissertation Committee Member

Approved:

_____ _____

Eric S. Ackerman, Ph.D. Date
Dean, Graduate School of Computer and Information Sciences

Graduate School of Computer and Information Sciences
Nova Southeastern University

2015

An Abstract of a Dissertation Submitted to Nova Southeastern University
in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

Efficient Variations of the Quality Threshold Clustering Algorithm

by
Frank Loforte, Jr.
March 2015

Clustering gene expression data such that the diameters of the clusters formed are no greater than a specified threshold prompted the development of the Quality Threshold Clustering (QTC) algorithm. It iteratively forms clusters of non-increasing size until all points are clustered; the largest cluster is always selected first. The QTC algorithm applies in many other domains that require a similar quality guarantee based on cluster diameter. The worst-case complexity of the original QTC algorithm is $O(n^5)$. Since practical applications often involve large datasets, researchers called for more efficient versions of the QTC algorithm.

This dissertation aimed to develop and evaluate efficient variations of the QTC algorithm that guarantee a maximum cluster diameter while producing partitions that are similar to those produced by the original QTC algorithm. The QTC algorithm is expensive because it considers forming clusters around every item in the dataset. This dissertation addressed this issue by developing methods for selecting a small subset of promising items around which to form clusters. A second factor that adversely affects the efficiency of the QTC algorithm is the computational cost of updating cluster diameters as new items are added to clusters. This dissertation proposed and evaluated alternate methods to meet the cluster diameter constraint while not having to repeatedly update the cluster diameters.

The variations of the QTC algorithm developed in this dissertation were evaluated on benchmark datasets using two measures: execution time and quality of solutions produced. Execution times were compared to the time taken to execute the most efficient published implementation of the QTC algorithm. Since the partitions produced by the proposed variations are not guaranteed to be identical to those produced by the original algorithm, the Jaccard measure of partition similarity was used to measure the quality of the solutions.

The findings of this research were threefold. First, the Stochastic QTC alone wasn't computationally helpful since in order to produce partitions that were acceptably similar to those found by the deterministic QTCs, the algorithm had to be seeded with a large number of centers ($ntry \approx n$). Second, the preprocessed data methods are desirable since they reduce the complexity of the search for candidate cluster points. Third, radius based methods are promising since they produce partitions that are acceptably similar to those found by the deterministic QTCs in significantly less time.

Acknowledgements

I would like to acknowledge the dissertation committee members Michael J. Laszlo, Ph.D. and Francisco J. Mitropoulos, Ph.D. for working with the dissertation chair Sumitra Mukherjee, Ph.D. in guiding the research of this dissertation. I would also like to acknowledge the help of Sharon L. Bear, Ph.D. and Susan L. Oliva, B.S. as editors that worked on this dissertation. Finally, I would like to acknowledge the support and editing that my wife, Renee Loforte, provided during the entire length of this dissertation.

Table of Contents

List of Tables

Tables

List of Figures

Figures

Chapter 1

Introduction

Background

Heyer, Kruglyak, and Yooseph (1999) developed the Quality Threshold Clustering (QTC) algorithm for gene clustering. The authors needed to find an analysis procedure that extracts useful clusters from newly available gene expression data. The QTC algorithm locates the largest clusters of open reading frames (ORFs), which are the parts of a gene that encode a protein, and satisfies a quality guarantee.

The inputs of the QTC algorithm are a set of points G in multidimensional space and a threshold value that represents the maximum diameter d of the clusters. Heyer et al. (1999) developed the following pseudo code to explain the QTC algorithm; Figure 1 is an excerpt from their paper.

1 Procedure $QT\_Clust(G, d)$
2 if ($|G| \leq 1$) then output G, else do /* Base case */
3 for each $i \in G$
4 set $flag = TRUE$; set $A_i = \{i\}$ /* A_i is the cluster started by i */
5 while (($flag = TRUE$) and ($A_i \neq G$))
6 find $j \in (G - A_i)$ such that *diameter* $(A_i \cup \{j\})$ is minimum
7 if (*diameter* $(A_i \cup \{j\}) > d$)
8 then set $flag = FALSE$
9 else set $A_i = A_i \cup \{j\}$ /* Add j to cluster A_i */
10 identify set $C \in \{A_1, A_2, \cdots, A_{|G|}\}$ with maximum cardinality
11 output C
12 call $QT\_Clust(G - C, d)$

Figure 1 – Pseudocode for the QTC algorithm

Given n points and a maximum allowable cluster diameter d, it iteratively forms clusters of similar maximum size until all points are clustered. Heyer et al. (1999)

suggested that a "termination criterion" could be added to stop the algorithm when the largest remaining cluster has less than a predefined number of data points (p. 1111). Visually inspecting the clusters at various thresholds preceded their choice of the threshold value. They concluded that the QTC algorithm, as compared to other clustering methods such as k-means, was better suited for their analysis of the data for two reasons.

First, Olson, Epstein, Sackett, and Yergey (2011) stated that varying the threshold of the QTC algorithm might change the size and number of clusters, but each cluster will have no unrelated patterns forced into it. This is known as the quality guarantee.

Second, algorithms that use a predetermined number of clusters, such as k-means, suffer from the following two conditions: a small number of clusters will cause unrelated patterns to be grouped together, and a large number of clusters will cause similar patterns to be split into separate groups (Heyer et al., 1999).

The QTC algorithm and its variations have been applied to various domains: The QTC algorithm was used by Yuan et al. (2008) in their placement algorithm for wireless sensor networks. Yaakob, Lim, and Jain (2009) used the QTC algorithm for pattern classification problems in the medical domain. Schafer and Fey (2008) created a new framework in grid computing using the QTC algorithm. Pukáncsik et al. (2010) applied the QTC algorithm to find the biggest cluster in their DNA research. The Stochastic QT-Clust algorithm, as developed by Scharl, Striedner, Pötschacher, Leisch, and Bayer (2009), is an adaptation of Heyer et al.'s (1999) original QT algorithm. The Stochastic QTC algorithm, which completes clustering in a fraction of the time of Heyer et al.'s algorithm, has been used to cluster microarray data sets. Finally, Danalis, McCurdy, and Vetter (2012) reduced the overall complexity of the original QTC algorithm from a

$O(n^5)$ to a $O(n^2)$ function. They accomplished this reduction by preprocessing the data prior to clustering so that the algorithm is not required to search all items during the cluster formation phase and by reducing the number of diameter calculations.

Problem Statement

The QTC algorithm yields clusters that satisfy a quality guarantee. Quality is quantified by the maximum acceptable cluster diameter. K-means and the other clustering methods do not provide this guarantee (Heyer et al., 1999). Danalis et al. (2012) analyzed the QTC algorithm and they concluded its overall complexity as $O(n^3 * F_D)$ where F_D is the complexity of the diameter function. The complexity of Heyer's diameter function is $O(n^2)$ and that makes the overall complexity $O(n^5)$.

Clustering problems often involve partitioning a large number of data points. For example, the QTC algorithm partitioned over 4,000 points into clusters in Heyer et al.'s (1999) research. Dan and Mocian (2009) applied the QTC algorithm to cluster documents within the context of web and text mining. They clustered similar news documents over a 24-hour period that amounted to between "100,000 and 200,000 news items" (p. 559). Due to these large data sets, researchers called for more efficient versions of the QTC algorithm (Dan & Mocian, 2009; Heyer et al., 1999).

Dissertation Goal

The goal of this dissertation was to develop modified versions of the QTC algorithm that are computationally more efficient than the original QTC algorithm. These modified versions identify clusters that are similar, if not identical, to those produced by the original QTC algorithm.

Specifically, this dissertation investigated methods for carefully selecting a small subset of k points around which to build each cluster ($k \ll |G|$), instead of constructing clusters for each point as in step 3 of Figure 1. The investigation includes a method of choosing these points and the evaluation includes experiments with various values of k.

This dissertation also investigated alternate methods to form the clusters instead of using the computationally expensive search to identify j, where j represents the candidate point that is evaluated for inclusion into the cluster that minimizes cluster diameter, as in step 6 of Figure 1. The investigation included two methods of forming clusters.

The measurable criterion of success was a modified QTC algorithm, which executed in significantly less time than the original QTC algorithm and aimed to produce similar partitions while using the same threshold diameter. The clusters produced are identical or very similar to the QTC algorithm's clusters. The partitions' similarity was evaluated using the "Jaccard Similarity on Entity Pairs" (Duan, Fokoue, Srinivas, & Byrne 2011). The Jaccard Similarity was well suited for this application because it was able to "capture the effect of big clusters" (p. 6). Big clusters are a definite possibility when using the QTC algorithm. The specified maximum cluster diameter threshold constraint was maintained. Finally, the proposed methods were evaluated using benchmark datasets found in the literature.

Research Questions

To address the problem statement and achieve the dissertation goal, the following research questions were used to guide the study:

Research question 1: What are some efficient and effective methods for selecting a small subset of k points around which to build the clusters?

Research question 2: What are some efficient and effective methods for identifying cluster neighborhoods?

Relevance and Significance

The QTC algorithm was developed with a quality guarantee for clustering genes. It has recently been applied to the organization of uracil-DNA-degrading factors (Pukáncsik et al., 2010) and for fixture detection in homes (Srinivasan, Stankovic, & Whitehouse, 2013).

The QTC algorithm and/or modified versions of it have been used in the following areas: image categorization (Ferecatu & Geman, 2009), gene expression (Jiang, Pei, & Zhang, 2005), market power potential (Lesieutre, Rogers, Overbye, & Borden, 2010), object identification (Nieto, 2010), mining documents for the Web (Dan & Mocian, 2009), and grid application componentization (Schafer & Fey, 2008).

The QTC algorithm does a computationally expensive search and is computationally much more demanding than k-means and other clustering algorithms (Dan & Mocian, 2009; Danalis, McCurdy, & Vetter, 2012). This renders the QTC algorithm impractical for use with large data sets. Since many applications today require clustering a large number of points with quality threshold guarantees, researchers called for more efficient versions of the QTC algorithm (Dan & Mocian, 2009; Heyer et al., 1999).

Preprocessing by partitioning the data points will result in a more efficient version of the QTC algorithm (Scharl & Leisch, 2006; Danalis et al., 2012). Further, the results can be generalized to other forms of clustering algorithms. Preprocessing by partitioning can be used to locate candidate clusters for various clustering algorithms. Application of

these methods can be used to improve the efficiency of other computationally expensive applications (Arthur & Vassilvitskii, 2007)

Barriers and Issues

The first barrier was to find a method that could decrease the expense of building clusters around every point. The QTC algorithm in step 3 of Figure 1 uses each point to create a cluster and then only uses the cluster with greatest cardinality. A modified process that uses a subset of points, instead of every point, reduced the total time of the algorithm.

The second barrier was to locate an efficient technique that could find points to include in the clusters. In step 6 of Figure 1 of the QTC algorithm every other point is checked to see if its inclusion into the cluster will cause the smallest increase in the diameter of the cluster. The modification of this method to only search a subset of the remaining points lowered the complexity of the algorithm.

One issue was to program the modified QTC algorithms to be efficient. Due to the $O(n^5)$ complexity of the QTC algorithm, inefficient code could have rendered the method impractical for use with large data sets.

Another issue was to select or generate the proper data sets to evaluate all the algorithms with data similar to what was used in the body of knowledge. Researchers are applying the QTC algorithm to data sets that range in cardinality from 5000 to 200,000 data points. Finding the proper cardinality for the test data sets provided good data to evaluate the pros and cons of the modified QTC algorithms.

Summary

The QTC algorithm uses a quality guarantee to find clusters that only include similar points; this is a criterion that some applications require. The quality guarantee is the maximum diameter of the clusters and it is the only input parameter needed besides the set of points. The algorithm has been identified as having an expensive search process by various researchers.

A more efficient QTC algorithm is needed by new applications that have a greater number of data points. Efficiency can be increased by selecting a small subsets of points to build the clusters around. Additionally, forming the clusters with a search that is not computationally expensive can increase efficiency.

Chapter 2

Review of the Literature

The literature review was broken into five sections. The sections are QTC Algorithms and its Variations, Applications of QTC, Computational Efficiency of QTC, Partition Similarity Measures, and Methods of Improving the QTC Algorithm.

QTC Algorithms and its Variations

The QTC clustering algorithm and its variants have been applied in several studies. Heyer et al. (1999) developed the original QTC algorithm for gene clustering. It is capable of locating the largest clusters of ORFs for genes. Their contribution was to create an algorithm that finds clusters without the information of how many clusters to locate. The algorithm requires a value for the diameter of the clusters as a termination criterion. The algorithm's quality guarantee ensures that each cluster will not have unrelated patterns forced into it.

Scharl et al. (2009) applied the Stochastic QT-Clust to microarray data sets and presented their results with neighborhood graphs. The Stochastic QT-Clust algorithm is an adaptation of Heyer et al.'s (1999) original QTC algorithm. In this adaptation, a parameter is added, the number $ntry$, which limits the number of clusters that are evaluated prior to selecting the largest cluster. Instead of forming clusters around all n data points, the clusters are formed starting from $ntry$ randomly selected data points (with $ntry \ll n$). A low value for $ntry$ will result in faster execution of the algorithm. However, an $ntry$ equal to the number of genes being evaluated will result in the same execution time as that of the original QTC algorithm.

Scharl and Leisch (2006) compared Heyer et al.'s (1999) QTC algorithm to k-means and the Stochastic QT-Clust, which adds the $ntry$ parameter. A yeast data set with 3722 genes was used for clustering. The $ntry$ parameter varied from 1 to 3300. They stated that outliers would not be part of any cluster, whereas k-means includes all genes, even outliers, in the clusters. They concluded that Stochastic QT-Clust is best when there is interest in small sums of within cluster distances. They stated that the original QTC algorithm is preferable if "stability and reproducibility" are important (p. 2).

Scharl and Leisch (2010) compared their Stochastic QT-Clust clustering algorithm to the k-means algorithm with both raw data and functional data of time course gene expression data; the functional data used curves fit to each observation to account for time dependency. The simulation study was performed by adding various types of noise to evaluate the performance of the algorithms. The results showed that QT-Clust outperformed k-means on both functional and raw data for low noise levels. However, k-means outperformed QT-Clust for medium and high noise levels.

Choudhury, Sarmah, and Sarma (2012) created a modified QTC algorithm to use in gene expression analysis. Instead of the jack-knife correlation coefficient that Heyer et al. (1999) used, they applied a modified version of Pearson's correlation coefficient. The original QTC algorithm was modified by adding a dynamically calculated minimum correlation value and creating the overall correlation factor (OCF). Their modified algorithm only needs one input parameter, minimum cluster size, and is able to calculate the other parameter dynamically. The OCF is used as a tie breaker when multiple clusters have the same number of genes and to detect high density clusters that exist inside of low density clusters.

Yaakob and Jain (2012) & (2009) combined QTC and the Fuzzy ARTMAP (FAM) architecture to analyze shape to classify insects. They modified the QTC algorithm by dynamically determining another parameter during the learning phase and set it according to characteristics in the data. The minimum cluster size was removed, which allowed for clusters of only one pattern. Finally, they used Euclidean distance as the similarity measure.

Applications of QTC

Olson et al. (2011) applied QTC to mass spectrometry data. QTC was chosen for its quality guarantee and because it does not require the number of clusters *a priori.* The algorithm yielded precision nodes that were proportionate to the instrument's mass measurement precision. Their application "uses replicate spectra and forms clusters of peaks within those spectra" (p. 970). The quality threshold was the mass measurement precision. It was previously validated for linear TOF (time of flight) measurements.

Pawlik, Alibert, Baulande, Vaigot, and Tronik-Le Roux (2011) used QTC to identify transcription factors that regulate early radiation response. A high correlation coefficient (0.8), of time-ordered gene expression profiles, for the quality threshold was used to arrive at high-quality clusters that had a reduced number of false negative predictions. They clustered 45,101 probe sets that corresponded to ~34,000 genes. The QTC algorithm was selected for its production of high quality clusters, that all possible clusters are considered, and the number of clusters was not needed prior to evaluation.

Geremek et al. (2011) applied the QTC algorithm to identify groups of genes that resulted in RNA expression patterns, which were highly correlated during the study of biopsies. The authors used 13,811 probes for clustering and identified 12 clusters with at

least 100 genes. They selected a cluster diameter of < 0.3 for the Jackknife correlation coefficients of the genes.

Danielson and Lill (2010) used QTC to group the solutions of protein loop regions. They used only 5,000 loop combinations in order to limit the computational time. The root-mean-square deviation value between solutions of each loop was their quality threshold, the maximum diameter was two angstroms.

Gu and Wang (2010) compared the QTC algorithm to k-means and random walks using tropical oceanic data. They used 1,200 samples that represented monthly averages for 100 years. The cross correlation of temperature and salinity was their data set. A list of thresholds, $\{d_0, d_1, d_2, \cdots, d_I\}$, were used with the QTC algorithm.

Croce, Giannone, Annesi, and Basili (2010) applied the QTC to latent semantic analysis (LSA). They used a set of LSA vectors that represent the frame elements of the semantic heads with 134,697 predicates and 271,560 arguments. Their quality threshold was set to 0.1, 0.5, and 0.85.

Dan and Mocian (2009) applied the QTC algorithm to document clustering within the context of web and text mining. Their system clusters between 100,000 and 200,000 news items every day. Their optimized algorithm only calculates the quality threshold, cosine similarity between documents, once as they are stored in a cache. Part of their algorithm has been modified to be incremental. They stated that, if they increase the number of sources their implementation might reach its limits.

Saito et al. (2009) used pairwise comparison as the first phase, QTC as the second phase, and k-means as the final phase to identify the clusters of gene expression patterns. During their time course evaluation of 5157 genes, they extracted 623 with a pairwise

comparison and then applied the QTC algorithm to identify 37 expression patterns, which k-means combined into eight clusters.

Hara, Ohnishi, and Horinouchi (2009) used QTC to predict the genes that were probable A-factor-inducible. They analyzed 477 genes that were expressed differently after the addition of A-factor. Those genes were grouped by QTC analysis according to expression patterns. From the resultant 15 groups, cluster 1 included the genes whose expression had increased, and cluster 2 was composed of the genes whose expression had decreased.

Coppe et al. (2009) used QTC to identify sets of co-expressed genes in myeloid cells. QTC of 2796 genes with a cluster diameter of 0.25 and at least 15 genes per cluster resulted in 44 gene clusters with a total of 2455 genes. They selected QTC for its ability to set the threshold for cluster quality and the number of genes per cluster. They performed the QTC with the MultiExperiment Viewer (MeV) software.

Bergholz et al. (2007) used QTC to cluster E. coli genes as Heyer et al. (1999) did using the same algorithm. The QTC algorithm placed 2,468 of the 2,552 ORFs into 12 groups. QTC was performed by the MeV v. 3.1 software at The J. Craig Venter Institute in La Jolla, California.

Minami, Maniratanachote, Katoh, Nakajima, and Yokoi (2006) applied the QTC algorithm to estimate the major gene expressions profiles based on ThioAcetamide (TA) dosage. Clustering was performed on a data set of 7978 genes with quality thresholds ranging from 0.82 to 0.92 for the correlation coefficients. They used the GeneSpring QTC algorithm and concluded that it is a “sensitive marker” that predicts potential hepatotoxicity (p. 64).

Minami et al. (2005) used the GeneSpring QTC algorithm to estimate the majority of the gene expressions profiles. They identified 17 potential toxicity markers by using QTC. A data set of 14,474 genes and quality threshold of 0.68 for the correlation coefficient. They concluded that the approaches of serum biochemical markers and two distinctive QTC evaluations yielded the same results.

Toledo-Rodriguez, Goodman, Illic, Wu, and Markram (2005) used the QTC algorithm to investigate gene expressions related to neocortical neurons. QTC was used as an unsupervised algorithm that was followed up with the supervised application of artificial neural network regression to predict the anatomical types present in the neurons. The authors experimented with various diameter thresholds and concluded that a diameter of 0.5 produced 7 clusters and provided "robust results" when they evaluated 268 neurons (p. 404).

Tanaka et al. (2011) applied the QTC algorithm to extract the clusters of upregulated (maltose-utilizing (LS) yeast) and downregulated (high-sucrose-tolerant (HS) yeast) genes under high-sucrose conditions with concentrations from 0 to 50%. Minimum cluster size was 100 and minimum correlation was 0.6. The first cluster included 225 upregulated genes and the second cluster had 124 downregulated genes, of a total of 1523 genes.

Sidorov, Hicks, Marshall, Sanei, and Chambers (2006) used QTC with their multi-camera 3D tracking system. They used four digital cameras connected to four computers. The 2D face positions are sent to a server that is running the QTC algorithm. The algorithm is applied to the "barycenters of all pairs of rays" originating from the cameras.

The output is the estimated 3D face coordinates and is relayed back to the 2D trackers for error correction.

Ferecatu and Geman (2009) applied the QTC algorithm to categorizing images in clusters based on similarities. This application of QTC is different than the other applications; instead of using a distance measure, they used individuals to match images based on similar features.

Jiang, Pei, and Zhang (2005) compared various algorithms to the Adaptive quality-based clustering (ADAPT) with gene expression data. The algorithms that they compared were: k-means, SOM, Cluster Affinity Search Technique (CAST), Cluster Identification via Connectivity Kernels (CLICK), Self Organizing Tree Algorithm (SOTA), Gene Pattern eXplorer (GPX), and ADAPT. When the results showed their method was not as efficient as some other clustering algorithms, they stated that biologists value the effectiveness over efficiency of the mining process.

Lesieutre, Rogers, Overbye, and Borden (2010) applied both the QTC and k-means algorithms to find the groups of generators that are "likely to be able to exercise market power". They used two steps to find the group of generators; first the generators were clustered, then the results were refined by evaluating the price perturbations. For the QTC algorithm they used Euclidean distance for the threshold and an optional maximum cluster size.

Nieto (2010) applied the QTC algorithm to determine the number of parts that are included in an object. The threshold considered the distance of the perceptual similarity between regions in an image. He also used a minimum of three items per cluster.

Pukáncsik et al. (2010) implemented the QTC algorithm in their DNA research. Their research looked for the biggest cluster with a minimal size of 25 and maximal root-mean-square deviation of 5 $\mathring{A}$. The QTC algorithm was used to cluster approximately 100,000 preliminary models.

Schafer and Fey (2008) used the original QTC algorithm as the first phase of their method. The second phase also used the QTC algorithm but added a diameter multiplier, which enlarged the maximum diameter by the multiplier. This second phase eliminated the possibility of having nodes that are not connected to other nodes; their method guaranteed a "complete hierarchy in every case" (p. 180).

Yuan et al. (2008) applied the QTC algorithm to their divide-and-conquer placement algorithm and demonstrated that clustering improved the placement of wireless sensors. They first used the QTC algorithm to cluster surveillance spots into groups by proximity. Then the constrained simulated annealing solver was used for each cluster to determine the location of the sensors. The number of surveillance spots was 225 regular spots and 200 random spots. The diameter used in the QTC algorithm was the impact region of the sensor.

Bednarik and Kovacs (2011) applied the Quality Threshold (QT) parameter, the threshold value for the maximum diameter d of the clusters, to the Hierarchical Agglomerative Clustering (HAC) algorithm. The authors performed an investigation to identify the clustering methods with an optimal level of similarity between the elements of the clusters. They stated that only the BIRCH (balanced iterative reducing and clustering using hierarchies) and the QTC algorithms used the radius of the clusters.

The authors stated that the "BIRCH algorithm defines a weak threshold on the volume" of clusters (p. 258). Whereas, the QTC algorithm's constraint is much stronger and was selected to be added to two special extensions of the HAC clustering method. They concluded that the best-first method was superior and based on the cost function the method should only be used with middle-sized problems.

Dutta & Overbye (2011) used the QTC algorithm to design a more efficient wind farm collector system cable layout than the conventional radial system cable layout. They applied the QTC over three levels to determine the most efficient layout possible. Then they combined the QTC method with the radial method for a hybrid version. Finally, they designed the conventional radial layout and compared the costs, power generated, and the reliability of the three versions.

The QTC version had the lowest cost, generated the most power, and was the most reliable of the three versions. They selected the QTC algorithm due to the quality guarantee which clusters similar objects together and did not require the number of clusters to be know beforehand.

Ha-Thuc, Nguyen, and Srinivasan (2008) used the quality threshold parameter of the QTC algorithm in their QT summarization algorithm. Their algorithm iterates through the k-means algorithm several times, at each iteration they use QTC to remove similar items. This is a novel approach that combines the QTC and k-means in order to remove the requirement of knowing the number of clusters *a priori.*

Tang, Zhang, Cheema, and Ressom (2010) applied the QTC algorithm to cluster peaks from the peak list that they created previously. They selected the peak candidates with higher intensity than the mean from the LC-MALDI-TOF (Liquid Chromatography-

Matrix-Assisted Laser Desorption/Ionization-Time Of Flight) run based on m/z (mass-to-charge ratio), RT (retention time), and intensity. Their method was useful when aligning the LC-MALDI-TOF runs from separate groups of samples.

DNA Nano array analysis is another research arena where the QTC has been applied. Waoo, Kashyap, and Jaiswal (2010) incorporated the QTC algorithm into their proposed Dynamically Growing Hierarchical Self Organizing Map (DGHSOM) to identify co-expressed genes. They presented 3 results at 50, 100, and 200 iterations of applying DGHSOM. The clusters increased from 9 to 19 as the iterations were increased. They concluded that when they increased the number of iterations of DGHSOM, their cancer diseases diagnosis results were better.

The discovery of electrical and water fixtures in a home is another application that used the QTC algorithm. The system called FixtureFinder developed by Srinivasan, Stankovic, and Whitehouse (2013) incorporates the data streams from infrared activity, ambient light levels, smart power, and water meters for the detection phase. The QTC algorithm was used to recognize patterns of multimodal pairs in the data stream. They deployed 25 to 40 sensors into 4 homes for 7 to 10 days of data collection. The results confirmed that the FixtureFinder system is able to identify 90% of the fixtures in a home in less than 10 days.

Computational Efficiency of QTC

Danalis et al. (2012) analyzed the QTC algorithm for its worst-case complexity. The overall complexity is composed of three parts, the 'for each' loop, the while loop, and the find operation. The find operation in line 6 of Figure 1 executes $O(n)$ times and calls the diameter function, its complexity is $O(n * F_D)$ where F_D is the complexity of the

diameter function used. The while loop, in line 5, executes until no more elements can be added to the cluster. It can have either a complexity of $O(n * F_D)$ for a data set with $O(n)$ clusters of $O(1)$ elements or $O(n^2 * F_D)$ for a data set with $O(1)$ clusters of $O(n)$ elements. The 'for each' loop which starts on line 3 executes $O(n)$ times. They concluded its overall complexity as $O(n^3 * F_D)$ for both extreme data sets.

They also recommend preprocessing the data prior to starting the QTC algorithm to reduce the complexity further. By creating a matrix of pair of elements which are within the threshold value of d, they can reduce the complexity to $O(n^2 * F_D)$. The complexity of the diameter function can also be reduced to $O(1)$ by using memory to store the largest distance of each element to the cluster and the last element added. Using memory to store the preprocessed matrix and the distances reduces the QTC algorithm's overall complexity to $O(n^2)$. The methodology section presents details of an adaptation of their method.

Partition Similarity Measure

Duan, Fokoue, Srinivas, and Byrne (2011) presented their "Measure I: Jaccard Similarity of Entity Pairs" as a method to quantify similarity of clusters which "captures the effect of big clusters in a partition" (p. 6). Their method generates all pairs of entities for each cluster in partitions P_1 and P_2. The result is the creation of the corresponding sets of entity pairs labeled P'_1 and P'_2. For example the partition $P_1 = \{\{a, b\}, \{c, d, e\}\}$ becomes $P'_1 = \{\{a, b\}, \{c, d\}, \{c, e\}, \{d, e\}\}$ and partition $P_2 = \{\{a, b, c\}, \{d, e\}\}$ becomes $P'_2 = \{\{a, b\}, \{a, c\}, \{b, c\}, \{d, e\}\}$. The similarity of the sets P'_1 and P'_2 is calculated with the standard Jaccard similarity where each entity pair is considered a basic element

of a set. The similarity of the partitions is computed using the formula $PSim_1(P_1, P_2) = \frac{|P'_1 \cap P'_2|}{|P'_1 \cup P'_2|}$. The similarity value's range is [0,1].

Methods of Improving the QTC Algorithm

The following research guided the development of the methods used to create the modified QTC versions in this dissertation:

Danalis, McCurdy, and Vetter (2012) analyzed the QTC algorithm and presented its worst-case complexity as $O(n^3 * F_D)$ where $F_D$is the complexity of the diameter function. When the complexity of the diameter function is $O(n^2)$ that makes the overall complexity $O(n^5)$. They recommend preprocessing the data prior to starting the QTC algorithm to reduce its complexity. By creating a matrix of pair of elements which are within the threshold value of d, they can reduce the complexity to $O(n^2 * F_D)$. After inserting the diameter function the overall complexity is reduced to $O(n^4)$. Some of the proposed modified QTC versions preprocessed the data by creating a matrix of the points.

Leisch (2006) compared the use of diameter and radius when evaluating clustering algorithms. He stated that the global minima of the two "will usually not be exactly the same" (p. 528). The research presented in the article demonstrates that the distance function used greatly affects the efficiency of the algorithm. He proposes that researchers should experiment with various standard distance measures to evaluate how they influence the algorithms. The introduction of the radius reduces the complexity to $O(n^2 * F_R)$ where F_R is the complexity of the radius function. Some of the proposed modified QTC versions used the radius of the cluster instead of the computationally expensive diameter function.

Chapter 3

Methodology

Overview

The goal of developing a modified QTC algorithm that is computationally more efficient than the QTC algorithm and identifies clusters that are similar, if not identical, to those produced by the original QTC algorithm was accomplished by implementing the original QTC algorithm, the Stochastic QTC algorithm, an adaptation of the Danalis QTC algorithm, and the various modified QTC versions using Visual C# as described in the Experimental Design section. These modified QTC versions were created using the strategies described in the sections 'Methods for Selecting a Subset of Points' and 'Methods for Identifying Cluster Neighborhoods'. All QTC versions were tested using the datasets described in the Benchmark Datasets section. The comparison of partition similarity for all QTC versions was measured with the method described in the Measure of Partition Similarity section. The efficiency of all QTC versions was measured with the method described in the Measure of Computational Efficiency section. Finally, the results were analyzed and presented as described in the Data Analysis section.

Experimental Design

Visual C# was used to create Heyer et al.'s (1999) algorithm, the Stochastic QTC algorithm, an adaptation of the Danalis et al.'s (2012) algorithm, and the proposed modified QTC algorithms for comparative evaluation. The Heyer algorithm was used to compare all the other algorithms with respect to cluster similarity, since the resulting clusters are always constant. The adaptation of the Danalis QTC algorithm was the

efficiency control. This allowed for the determination of the effectiveness of the proposed modified QTC algorithms.

Coding of the QTC Methods

Each of the following QTC methods were coded with the convention XYZ; where

X = 'D' if deterministic and 'S' if stochastic,

Y = 'd' if diameter based and 'r' if radius based, and

Z = 'P' if the data was preprocessed and 'N' if not.

Method for Selecting a Subset of Points

An alternate method for selecting initial cluster centers that avoid the exhaustive search of the original QTC algorithm led to faster execution times for the proposed modified QTC algorithms. The Stochastic QTC algorithm uses the parameter $ntry$ to limit the initial cluster search, Scharl et al. (2009). Instead of trying each point in during the cluster search, they only evaluate a random set of $ntry$ points. This lessened the impact of the exhaustive search that the Heyer QTC algorithm used.

Stochastic QTC (SdN) Algorithm

The Stochastic QTC (SdN) algorithm is an adaptation of Heyer's QTC (DdN) algorithm that adds the parameter $ntry$, Scharl et al. (2009). They use this parameter to limit the exhaustive search for the initial cluster points. Figure 2 is the pseudocode for the Stochastic QTC (SdN) algorithm. The difference between the Heyer QTC (DdN) algorithm and the Stochastic QTC (SdN) algorithm is the addition of line 2.1 and the modification of line 3 in Figure 2.

1 Procedure $QT\_Clust\_Stochastic(G, d, ntry)$
2 if ($|G| \leq 1$) then output G, else do /* Base case */
2.1 $S = random\_sample\_without\_replacement(G, \min(|G|, ntry))$
3 for each $i \in S$
4 set $flag = TRUE$; set $A_i = \{i\}$ /* $A_i$is the cluster started by i */
5 while (($flag$) and ($A_i \neq G$))
6 find $j \in (G - A_i)$ such that *diameter* $(A_i \cup \{j\})$ is minimum
7 if (*diameter* $(A_i \cup \{j\}) > d$)
8 then set $flag = FALSE$
9 else set $A_i = A_i \cup \{j\}$ /* Add j to cluster A_i */
10 identify set $C \in \{A_1, A_2, \cdots, A_{|G|}\}$ with maximum cardinality
11 output C
12 call $QT\_Clust\_Stochastic(G - C, d, ntry)$

Figure 2 – Pseudocode for the Stochastic QTC (SdN) algorithm

- $random\_sample\_without\_replacement(G, m)$: Returns m random points of G.

Modified QTC 1 (DrN)

This algorithm used the Heyer QTC (DdN) algorithm but instead of diameter of the cluster, it used the radius from a central point. The radius was calculated as $r = d/2$. These modifications are in lines 6 and 7 of Figure 3.

1 Procedure $QT\_Clust\_radius(G, d)$
2 if ($|G| \leq 1$) then output G, else do /* Base case */
3 for each $i \in G$
4 set $flag = TRUE$; set $A_i = \{i\}$ /* $A_i$is the cluster started by i */
5 while (($flag$) and ($A_i \neq G$))
6 for each $j \in (G - A_i)$
7 if (*distance* $(i, j) > d/2$)
8 then set $flag = FALSE$
9 else set $A_i = A_i \cup \{j\}$ /* Add j to cluster A_i */
10 identify set $C \in \{A_1, A_2, \cdots, A_{|G|}\}$ with maximum cardinality
11 output C
12 call $QT\_Clust\_radius(G - C, d)$

Figure 3 – Pseudocode for the QTC 1 (DrN) algorithm

Modified QTC 2 (SrN)

This algorithm modified the Stochastic QTC (SdN) algorithm to use radius instead of diameter. These modifications are in lines 6 and 7 of Figure 4.

1 Procedure $QT\_Clust\_Stochastic\_Radius(G, d, ntry)$
2 if ($|G| \leq 1$) then output G, else do /* Base case */
2.1 $S = random\_sample\_without\_replacement(G, ntry)$
3 for each $i \in S$
4 set $flag = TRUE$; set $A_i = \{i\}$ /* A_i is the cluster started by i */
5 while (($flag$) and ($A_i \neq G$))
6 find $j \in (G - A_i)$ such that *distance* (i, j) is minimum
7 if (*distance* $(i, j) > d/2$)
8 then set $flag = FALSE$
9 else set $A_i = A_i \cup \{j\}$ /* Add j to cluster A_i */
10 identify set $C \in \{A_1, A_2, \cdots, A_{|G|}\}$ with maximum cardinality
11 output C
12 call $QT\_Clust\_Stochastic\_Radius(G - C, d, ntry)$

Figure 4 – Pseudocode for the QTC 2 (SrN) algorithm

Methods for Identifying Cluster Neighborhoods

Identifying neighborhoods by using the selected center points and a small set of carefully selected points as done by Danalis et al. (2012), reduced the amount of calculations performed to form clusters. Only points near to the center point were considered. They used a procedure to select the subset of points for cluster formation that were within distance d of the initial cluster point, Figures 6 and 7 are the pseudocodes of their method. Priority queues M_i and M_j were used to store the points that were within distance (d) of the points indexed by i and j respectively, and stored in non-decreasing order of distances.

```
Procedure PreprocessData(G, d)     /* G is a set of points, d is the diameter */
    n = |G|
    initialize(M_i)     /* M_i is initialized */
    initialize(M_j)
    for i = 1 to n − 1
        for j = i + 1 to n
            d_ij = distance( G_i, G_j)
            if  d_ij ≤ d
                insert(M_i, (j, d_ij))  /* store j and the distance, d_ij , to i in non-
                                           decreasing order of distances */
                insert(M_j, (i, d_ij))
```

Figure 5 – Pseudocode for the Preprocess Data procedure

Danalis QTC (DdP) Algorithm

The Danalis QTC (DdP) adaptation uses the preprocess data procedure. Its application reduced the $O(n^5)$ complexity of the Heyer's QTC algorithm to $O(n^4)$. The Danalis QTC (DdP) adaptation and three of modified QTC versions incorporate the preprocess data procedure. The modifications made for the Danalis QTC (DdP) adaptation are in lines 1.1 and 6 in Figure 8.

```
1   Procedure QT_Clust(G, d)
1.1 PreprocessData(G, d)
2   if (|G| ≤ 1) then output G, else do    /* Base case */
3     for each i ∈ G
4       set flag = TRUE; set A_i = {i} /* A_i is the cluster started by i */
5       while ((flag) and (A_i ≠ G))
6         find j ? M_i − A_i such that diameter (A_i ? {j}) is minimum
7         if (diameter (A_i ? {j}) > d)
8           then set flag = FALSE
9           else set A_i = A_i ∪ {j} /* Add j to cluster A_i */
10  identify set C ∈ {A_1, A_2, ⋯, A_|G|} with maximum cardinality
11  output C
12  call QT_Clust(G − C, d)
```

Figure 6 – Pseudocode for an adaptation of the Danalis QTC (DdP) algorithm

QTC Algorithms

Deterministic Methods				
No	Method	Stochastic	Uses Radius	Preprocesses Data
1	Heyer QTC (DdN)	No	No	No
2	Danalis QTC (DdP)	No	No	Yes
3	QTC 1 (DrN)	No	Yes	No
4	QTC 4 (DrP)	No	Yes	Yes
Stochastic Methods				
5	Stochastic QTC (SdN)	Yes	No	No
6	QTC 3 (SdP)	Yes	No	Yes
7	QTC 2 (SrN)	Yes	Yes	No
8	QTC 5 (SrP)	Yes	Yes	Yes

Table 1 – Comparison of All QTC Algorithms by Methods

Stochastic in column 3 of Table 1 implies that the clusters are built around a subset of points that are randomly chosen. The size of the subset of points can be varied using the $ntry$ parameter, Scharl et al. (2009). A smaller $ntry$ parameter will provide a more efficient algorithm, but will cause less similarity of clusters to the Heyer QTC (DdN) algorithm. Various increasing $ntry$ parameters were tried to find when the Jaccard Similarity value was at least 0.9 for the dataset and algorithm combinations that were tested.

The QTC 1 (DrN) algorithm simply replaced distance with radius. The QTC 2 (SrN) algorithm combined the Stochastic QTC method with the radius method. The QTC 3 (SdP) combined the Stochastic QTC method with the Danalis preprocess data method, Danalis et al. (2012). The QTC 4 (DrP) combined the Radius method with the preprocess data method. The QTC 5 (SrP) combined the Stochastic QTC with the Radius and the Preprocess data methods.

Benchmark Datasets

The following datasets were used for experimental evaluation of the algorithms. The sizes of these datasets range over most of the applications in which QTC has been used.

The TSP-LIB-1060 dataset has a cardinality of 1,060 two-dimensional points. The dataset is at the low end of the spectrum and it provided a benchmark for the baseline of the original QTC algorithm. It was labeled as Dataset 1. Various diameter/radius values were tested to provide the best Jaccard similarity values. The best similarity value results were obtained with a diameter of 2k and radius of 1k.

The SIM dataset's cardinality, 10,000 three-dimensional points, is slightly greater than most of the datasets that have been used by the researchers that use the QTC algorithm. The SIM dataset was provided and used by Laszlo & Mukherjee (2006). It was labeled as Dataset 2. Various diameter/radius values were tested to provide the best Jaccard similarity values. The best similarity value results were obtained with a diameter of 0.1 and radius of 0.05.

Measure of Partition Similarity

The resulting data clusters were evaluated using the Jaccard similarity measure. The "Jaccard Similarity on Entity Pairs," as presented by Duan et al. (2011), quantifies the similarity of clusters when comparing two partitions. Their method generates all pairs of entities in P_1 and P_2. The corresponding sets of entity pairs are P'_1 and P'_2. The similarity of the sets P'_1 and P'_2 is calculated with the standard Jaccard similarity, where each entity pair is considered a basic element of a set. The similarity of the partitions is computed using the formula $PSim_1(P_1, P_2) = \frac{|P'_1 \cap P'_2|}{|P'_1 \cup P'_2|}$. The similarity value's range is $[0,1]$.

Measure of Computational Efficiency

The computational efficiency was presented as a ratio of the time of execution between the Danalis QTC adaptation (DdP) and all the other QTC algorithm versions for each dataset and diameter combination. All execution times will be recorded on the same computer and will be an average of five runs.

Data Analysis

The analysis of the results was presented in a table similar to Table 2 to show the differences between all of the QTC algorithm versions with respect to cluster diameters, $ntry$ values, and the data set cardinality, Scharl et al. (2009). The fields of the table are the name of the QTC algorithm version, $ntry$ value, mean execution time, efficiency ratio when compared to the Danalis QTC adaptation (DdP), and Jaccard similarity value when compared to the Heyer QTC algorithm (DdN). The data analysis was evaluated to determine if the research questions were answered and the algorithms were ranked to identify the most efficient.

QTC Algorithm	$ntry$	Mean Execution Time	Efficiency Ratio	Jaccard Similarity Value
Heyer QTC (DdN)				1
Danalis QTC (DdP)			1	
QTC 1 (DrN)				
QTC 4 (DrP)				
Stochastic QTC (SdN)				
QTC 3 (SdP)				
QTC 2 (SrN)				
QTC 5 (SrP)				

Table 2 – Analysis of the Comparison Results of All QTC Algorithms Tested by Dataset

Resources

The following resources were needed to complete the dissertation:

- Computer hardware: MacBook Pro, iMac.
- Computer software: R language (R for Mac OS X), flexclust package in R, C#, Sleipnir Library, Pycluster Library, NumPy package, C Clustering Library, Microsoft Word, Microsoft Excel, Microsoft Visual Studios, EndNote, Safari.
- Networks: NSU Library, GSCIS, DTS.

Chapter 4

Results

This chapter includes three sections that describe the Data Analysis, Findings, and Summary of Results. The data analysis section presents the how the data was collected and how it was analyzed. The findings section explains what was discovered during analyzing the data. The summary of results section lists the results that were derived from the data.

Data Analysis

The results in Tables 3 and 4 represent the data collected from all of the experiments performed. Each algorithm was tested using two datasets and five subsets of Dataset 2. Dataset 1 had 1,060 data points and Dataset 2 had 10,000 data points. Five subsets were also created from Dataset 2 to evaluate the complexity of the algorithms. The subsets ranged from 1,000 to 5,000 data points in increments of 1,000.

Ten cluster diameters were evaluated to identify the best threshold value for the datasets used. Ten values of $ntry$ were evaluated with all the algorithms that included the Stochastic method, to find the Jaccard Similarity values of at least 0.9. Then each algorithm was run with the diameter and $ntry$ values identified above, five times to calculate an average running time in seconds. These results are in the third column of Tables 3 and 4.

The Efficiency Ratio, in column four, is a ratio of the running time of the algorithm under consideration over the running time of the Danalis QTC adaptation (DdP). The

Danalis QTC adaptation (DdP) was the most efficient of the three published algorithms and was used as the control.

Each algorithm-dataset-diameter experiment also produced a file that listed which data points were clustered together. These files were then compared with the files that the Heyer algorithm (DdN) produced using the Jaccard Similarity of Entity Pairs method. The results are the Jaccard Similarity Values in the fifth column of both tables.

The Efficiency Ratio results were analyzed to identify which algorithms were more efficient than the Danalis QTC adaptation (DdP). The highlighted results are those that are less than one and represent a result that is more efficient than the control.

The Jaccard Similarity Value results were also analyzed to identify which algorithms produced clusters that were the most similar to those produced by the Heyer algorithm (DdN). Values that are the closest to the value of one are the most similar to the control. The Danalis QTC adaptation (DdP), a deterministic method, produced clusters that were identical to the Heyer algorithm (DdN).

The variations in execution time of those algorithms that incorporate the non-deterministic Stochastic method were analyzed in Tables 5 and 6. The tables have columns for the algorithm name, mean, maximum, minimum, and standard deviation for the five run times.

The analysis of the five subsets of Dataset 2 allowed for an empirical measurement of the complexity of all the QTC algorithms evaluated. The trendlines shown in Charts 3 thru 9 demonstrates how a linear increase to the cardinality of the data points affects each of the algorithms. Each Chart plots that data points in thousands on the x-axis with the number of seconds on the y-axis. The trendlines were calculated using Microsoft Excel

and it provided the equations that were used to extrapolate the run times, which exceeded 12 hours with Dataset 2.

Dataset 1 – 1060 points QTC Algorithm	$ntry$	Mean Execution Time-sec.	Efficiency Ratio	Jaccard Similarity Value
Heyer QTC (DdN)	na	363	4.84	1
Danalis QTC (DdP)	na	75	1	1
QTC 1 (DrN)	na	1	0.013	0.93
QTC 4 (DrP)	na	0.4	0.005	0.93
Stochastic QTC (SdN)	850	327	4.36	0.94
QTC 3 (SdP)	850	65	0.87	0.95
QTC 2 (SrN)	850	0.1	0.001	0.9
QTC 5 (SrP)	850	0.1	0.001	0.93

Table 3 – Analysis of the Results of All QTC Algorithms Tested for Dataset 1

The results of Dataset 1 in Table 3 demonstrated that both the Stochastic and Danalis methods improved on the Heyer QTC algorithm, but Danalis QTC adaptation (DdP) was the best at reducing the time to run the QTC. When combining the Stochastic and Danalis methods with the radius method, it was found that the Stochastic method complemented and enhanced the radius method the most and lowered the Efficiency Ratio of QTC 1 (DrN) vs. QTC 2 (SrN). The Danalis method also complemented the radius method, and increased its efficiency but resulted in a less improved Efficiency Ratio of QTC 1 (DrN) vs. QTC 4 (DrP). When the Danalis method was combined with the Stochastic method, QTC 3 (SdP) and QTC 5's (SrP) Efficiency Ratios both improved.

Chart 10 shows the scatterplot of Dataset 1 with the first cluster and the diameter threshold identified with a red circle. The scatterplot demonstrates the dispersion of data points and it can be clearly seen that there are no circular clusters.

Dataset 2 – 10k points QTC Algorithm	$ntry$	Mean Execution Time-sec.	Efficiency Ratio	Jaccard Similarity Value
Heyer QTC (DdN)	na	*1,692,545*	19.2	1
Danalis QTC (DdP)	na	*88,296*	1	0.94
QTC 1 (DrN)	na	122	0.001	0.94
QTC 4 (DrP)	na	304	0.003	0.94
Stochastic QTC (SdN)	8500	*1,656,558*	18.8	0.98
QTC 3 (SdP)	8500	*81,192*	0.92	0.94
QTC 2 (SrN)	8500	122	0.001	0.98
QTC 5 (SrP)	8500	325	0.004	0.94

Table 4 – Analysis of the Results of All QTC Algorithms Tested for Dataset 2

The results of Dataset 2 in Table 4 demonstrated similar results to those found in Table 3 with one interesting difference. The higher cardinality of Dataset 2 identified that when the Danalis preprocess data method is combined with the Radius method the result is less efficient than the Radius method alone. It was also noted that the Stochastic method did not improve the Radius method when there was a higher cardinality of data points.

Chart 11 shows the scatterplot of Dataset 2 with the first cluster identified with blue marks. The scatterplot demonstrates the dispersion of data points and it can be clearly seen that it has circular clusters. The QTC algorithm was designed to identify circular clusters, so it works well with datasets like Dataset 2.

The Mean Execution Times in italics indicate that the values were extrapolated from the equations found on Charts 5 thru 9. These execution times were longer than 12 hours and could not be run repeatedly like the other times noted in Table 4.

Dataset 1 – 1060 points QTC Algorithm	Mean (seconds)	Maximum (seconds)	Minimum (seconds)	Standard Deviation
Stochastic QTC (SdN)	327.4	334	320	5.3
QTC 2 (SrN)	0.1	0.11	0.09	0.01
QTC 3 (SdP)	65	66	63	1.4
QTC 5 (SrP)	0.1	0.13	0.08	0.02

Table 5 – Analysis of the Run Time Results of All Non-Deterministic QTC Algorithms Tested for Dataset 1

Dataset 2 – 10k points QTC Algorithm	Mean	Maximum	Minimum	Standard Deviation
Stochastic QTC (SdN)	1,656,558	1,656,680	1,656,400	102
QTC 2 (SrN)	122	130	118	4.7
QTC 3 (SdP)	81,192	81,229	81,098	54
QTC 5 (SrP)	325	331	320	5.0

Table 6 – Analysis of the Run Time Results of All Non-Deterministic QTC Algorithms Tested for Dataset 2

Findings

Computational Efficiency

The highlighted results in the Efficiency Ratio column in Tables 3 and 4 were used to identify the most efficient QTC algorithm. All of the modified QTC algorithms resulted in running times that were more efficient than the Danalis QTC (DdP) algorithm for both datasets. QTC 2 (SrN) had the best overall running times and was the most efficient algorithm. The other efficient algorithm was QTC 1 (DrN).

All of the QTC algorithms were placed in Charts 1 and 2 and ranked from the longest running time to the shortest for Dataset 2. Starting with the Heyer algorithm on the left side of the chart and decreasing running time going to the right. The run times were shown in $\log_{10}(\text{seconds})$ to better focus on the differences of all the algorithms. The QTC algorithms were ranked in the following order based on the results; Heyer (DdN),

Stochastic (SdN), Danalis (DdP), QTC 3 (SdP), QTC 4 (DrP), QTC 5 (DrP), QTC 1 (DrN), and QTC 2 (SrN).

The QTC algorithms running times were also graphed with increasing cardinality to evaluate the empirical complexity in Charts 3 thru 9. Chart 3 shows that the Heyer QTC (DdN) Algorithm's empirical complexity was $O(n^4)$ where the theoretical asymptotic analysis is $O(n^5)$. This indicates that the complexity of the diameter function used, F_D from the equation $O(n^3 * F_D)$, was not as complex as Danalis' worst case complexity analysis.

Chart 4 demonstrates that the Danalis QTC (DdP) Algorithm's empirical complexity was also $O(n^4)$, but it did improve efficiency by a factor of two over the Heyer QTC (DdN) Algorithm. Chart 5 reveals that the Stochastic QTC (SdN) Algorithm's empirical complexity was only slightly better than the Heyer QTC (DdN) Algorithm. This was primarily due to the high percentage of search points that was required to maintain an acceptable cluster similarity.

Chart 6 establishes that the QTC 1 (DrN) and QTC 2 (SrN) Algorithms' empirical complexity were the best at $O(n^2)$. This was attributed to the less complex radius method that does not have to repetitively calculate the diameters as in the Heyer QTC (DdN) Algorithm. Chart 7 shows that the QTC 3 (SdP) Algorithm's empirical complexity was on par with the Danalis QTC (DdP) Algorithm's complexity with only a slight improvement.

Chart 8 demonstrates that the QTC 4 (DrP) Algorithm's empirical complexity was almost $O(n^3)$, this was due to the inclusion of the Danalis preprocessing data method which was affected by the large cardinality datasets. Chart 9 reveals that the QTC 5 (SrP)

Algorithm's empirical complexity was similar to QTC 4 (DrP) Algorithm's complexity, but it was slightly less efficient with the addition of the Stochastic method.

Summary of the findings:

Stochastic QTC alone isn't computationally helpful since in order to produce partitions that are acceptably similar to those found by deterministic QTC, the algorithm has to be seeded with a large number of centers ($ntry \approx n$).

Preprocessing data before the QTC algorithm runs is desirable when possible, for it lessens that exhaustive search for the cluster neighborhood points. However, the computer resources must be high so that a bottleneck does not happen.

Radius based methods are promising since they produce partitions that are acceptably similar to those found by deterministic QTC in significantly less time.

Summary of Results

The experiments and results conclusively answered both research questions that guided this research. The first question asked, "What are some efficient and effective methods for selecting a small subset of k points around which to build clusters?"

The answer is the Stochastic QTC method selected smaller sets of k points to build clusters around. This made the algorithm slightly more efficient than Heyer's QTC algorithm. The Stochastic method was also used in QTC 2 (SrN), QTC 3 (SdP), and QTC 5 (SrP).

The second question was, "What are some efficient and effective methods for identifying cluster neighborhoods?" The answer is both the Danalis QTC adaptation (DdP) and the use of radius instead of diameter made for efficient cluster formation. QTC

3 (SdP), QTC 4 (DrP), QTC 5 (SrP), and the Danalis QTC adaptation (DdP) showed better efficiency when compared to the Heyer algorithm with Datasets 1 and 2.

The use of radius was another method that efficiently identified cluster neighborhoods. The QTC 1 (DrN), QTC 2 (SrN), QTC 4 (DrP), and QTC 5 (SrP) all resulted in better efficiency than the Heyer (DdN), Danalis QTC adaptation (DdP), and the Stochastic QTC (SdN) algorithms.

The combination of the Stochastic method and radius, QTC 2 (SrN), resulted in the most efficient algorithm. QTC 2 (SrN) combines the Stochastic method and radius; this combination was the most efficient of all the algorithms tested. This efficiency can also be seen in the equation of Chart 6 where it empirically demonstrates that its complexity is much less than the other algorithms.

Chapter 5

Conclusions, Implications, Recommendations, and Summary

Conclusions

The data collected from the experiments provided results that were analyzed to identify the most efficient algorithms, which were the Stochastic combined with the radius method and the radius method alone. The combination of both of these methods yielded the most efficient proposed algorithm, the QTC 2 (SrN). Chart 1 and 2 demonstrate this clearly where the QTC 2 algorithm is at the far right side. This conclusively shows that the Stochastic method is more efficient than the Heyer method and the radius method is also more efficient than the Danalis preprocessing data method.

All of the methods and combinations improved on Heyer's QTC (DdN) algorithm and are useful when certain criteria are needed for the results. When efficiency is most important then the Stochastic/radius combination of QTC 2 (SrN) will yield the lowest running time. When consistency of the resulting clusters is required the QTC 1 (DrN) algorithm will be the best choice, because it does not use the random selection of points that the Stochastic method incorporates and its clusters will be the same from run to run.

All of the methods and combinations also exhibited no appreciable difference with respect to cluster similarity when compared to the clusters of the Heyer algorithm. The Jaccard Similarity Values were consistent across all algorithms, which means there is no reason to select one method over another solely based on cluster similarity.

Implications

This research proves that there are more efficient versions of the QTC algorithm that can be used on large datasets, which is exactly what researchers are asking for. The results that were generated quantify the extent that each algorithm will impact the efficiency on both dataset cardinalities. Researchers can now use these new methods to improve the QTC algorithms they create for the myriad of applications.

Recommendations

Future research should study the Danalis preprocessing data method further with large datasets, but they should use a computer that has enough memory that will not cause a bottleneck and adversely affect the efficiency. The Danalis method did improve the efficiency of the lower cardinality dataset and with the proper environment for the higher cardinality datasets; it may well be a contender for an efficient algorithm especially if it is combined with the radius method. The Danalis article also presented a method to reduce the complexity of the diameter function by using additional memory; this method may also prove to provide for an increase in efficiency. The radius method reduces the complexity of the algorithm and it can be easily combined with the Danalis preprocessing data method to increase its efficiency.

Summary

Clustering gene expression data such that the diameters of clusters formed are no greater than a specified threshold was what prompted the creation of the original QTC algorithm. Heyer et al. accomplished this goal in 1999, however the algorithm had a complexity of $O(n^5)$ and would only be practical for datasets with small cardinality. As

the datasets increase in cardinality the running time of the algorithm becomes prohibitive. This problem prompted many researchers to develop modified QTC algorithms that have less complexity and can work with the larger cardinality datasets.

The goal of this dissertation was to develop modified versions of the QTC algorithm that are computationally more efficient than Heyer's algorithm, yet find clusters that were similar if not identical to those of the Heyer algorithm. Five modified versions of the QTC algorithm were developed to evaluate the efficiency of several methods that promised to improve the efficiency of the algorithm. Those methods are the Danalis preprocessed data, Stochastic $ntry$ parameter, and the use of radius instead of diameter to find the clusters in the QTC algorithm.

The Danalis preprocessed data method aims to reduce the complexity of the QTC algorithm by finding all of the points that are within the diameter distance of every point and storing that information in a matrix. The matrix is used in the algorithm to reduce the extensive search for candidate points for the cluster formation. The preprocessed data method may be more effective than the other methods, if there is sufficient computer memory to allow the method to preform properly.

The Stochastic $ntry$ parameter method reduces the complexity of the QTC algorithm by only evaluating a subset of points during the search for the largest cluster. The $ntry$ parameter can be varied to allow for different size subsets for the search, Scharl et al. (2009). This research used an $ntry$ parameter that was 85% of the cardinality of Datasets 1 and 2. At that percentage the increase in efficiency was marginal when compared with the Danalis preprocessed data method. Finding methods for judiciously selecting a smaller set of points while maintaining cluster similarity is an area of future research.

The radius method uses the radius instead of the diameter when finding points that can be used when creating the clusters. This method reduces the complexity by removing the inefficient calculations to determine if a point increases the cluster diameter by the least amount. The radius method uses all points that are found to be less than or equal to the distance of the radius and does not reevaluate those that are further from the center point.

It was expected that the radius method might affect the similarity of the clusters due to the use of a center point and not calculating the diameter. When there are only a few points the requirement of using a center point could cause more clusters of smaller size, instead of less clusters of larger size when using the diameter. The results did not exhibit a difference in cluster similarity when the radius method was used, therefore the perceived effect was minimal and was not significant.

Eight algorithms were programmed to evaluate the QTC algorithms that were developed to address the inefficiency issue that Heyer's original QTC algorithm exhibited. The Heyer QTC (DdN), Danalis QTC adaptation (DdP), and Stochastic QTC (SdN) algorithms were coded to document their efficiency and use their results as controls for this research. Additionally, five other modified QTC algorithms were created to evaluate the interaction of the methods listed earlier.

The QTC 1 (DrN) algorithm replaced the diameter with the radius; this reduces the complexity of the algorithm by removing the inefficient calculations to determine if a point increases that diameter by the least amount. The radius was calculated as $r = d/2$. This algorithm fell on the right half of Charts 1 and 2, which was the side with the most efficient algorithms.

The QTC 2 (SrN) algorithm combined the Stochastic method and the radius method to evaluate the efficiency of their combination. This proved to be the best of all the modified QTC algorithms; this was the case with both datasets showing better efficiency in Charts 1 and 2. These two methods complemented each other and did not deter from the other method's strengths.

The QTC 3 (SdP) algorithm combined the Stochastic method with the Danalis preprocess data method. The combination of these methods produced an efficient algorithm for the lower cardinality dataset, but suffered on the dataset with the higher cardinality. It was also observed that the addition of the preprocess data method caused the Stochastic method to become less efficient.

The QTC 4 (DrP) algorithm combined the radius method with the Danalis preprocess data method. The combination of these two methods improved on the efficiency of the Heyer algorithm and fell in the middle of the pack in Chart 1. It was also observed that the addition of the Danalis preprocess data method caused the radius method to become less efficient.

The QTC 5 (SrP) algorithm combined the Stochastic method with the radius method and the Danalis preprocess data method. This algorithm combined three methods to produce the third most efficient algorithm when used with the larger cardinality dataset. Even tough it placed well in Chart 1; it is evident that the inclusion of the Danalis preprocess data method caused the other two methods to suffer less efficiency. This can be seen when QTC 5 (SrP) is compared with QTC 2 (SrN).

Both research questions were answered conclusively. The first research question sought efficient and effective methods to select a small subset of k points around which

to build the clusters. The Stochastic method was efficient and effective at selecting a smaller subset of k points. It was found that to achieve acceptable cluster similarity values, the percentage of points in the subset had to be 85%. The Stochastic method ended up being part of the most efficient modified QTC algorithm when combined with the radius method.

The second research question sought efficient and effective methods for identifying cluster neighborhoods. The Danalis preprocess data and the radius methods both were efficient and effective at identifying cluster neighborhoods. Even though the Danalis preprocess data method improved on the efficiency of the Heyer algorithm it was negatively affected by the dataset with higher cardinality. The radius was the most efficient and it resulted in being part of the two most efficient modified QTC algorithms along with the Stochastic method.

The goal of this dissertation was achieved by developing five modified versions of the QTC algorithm that were computationally more efficient that the original QTC algorithm. These modified versions also identified clusters that were similar if not identical to those produced by the original QTC algorithm. These results can be verified in Tables 3 and 4 and in Charts 1 and 2.

Researchers that wanted to use the QTC algorithm yet shied away form it due to its inefficiency can now use any of the modified QTC algorithms included in this research to improve on the performance of Heyer's QTC algorithm. Future research should strive to evaluate the Danalis preprocess data method with sufficient computer resources to determine if it can improve on these results.

Appendices

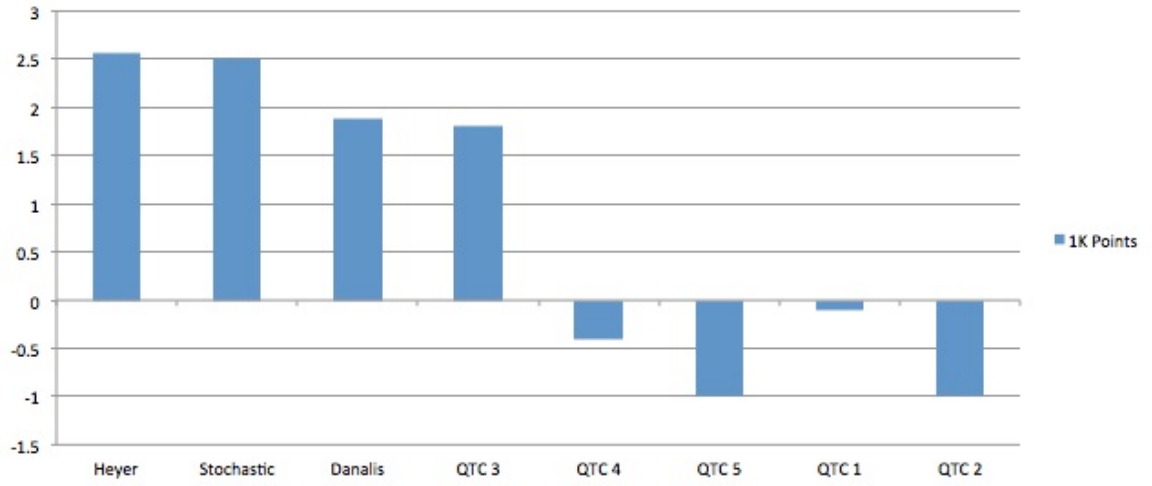

Chart 1 – Running Times in $\log_{10}$(seconds) by Algorithms for Dataset 1

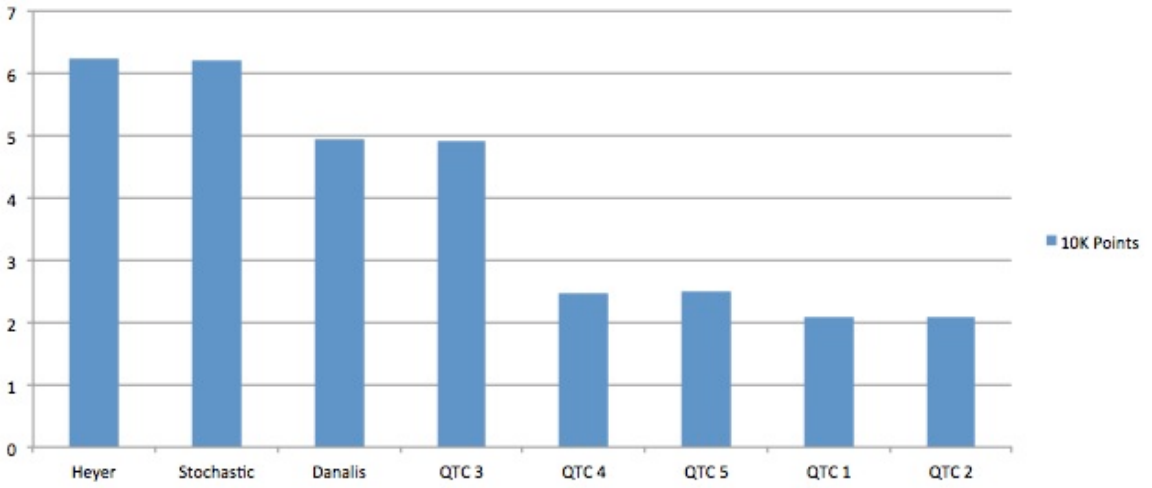

Chart 2 – Running Times in $\log_{10}$(seconds) by Algorithms for Dataset 2

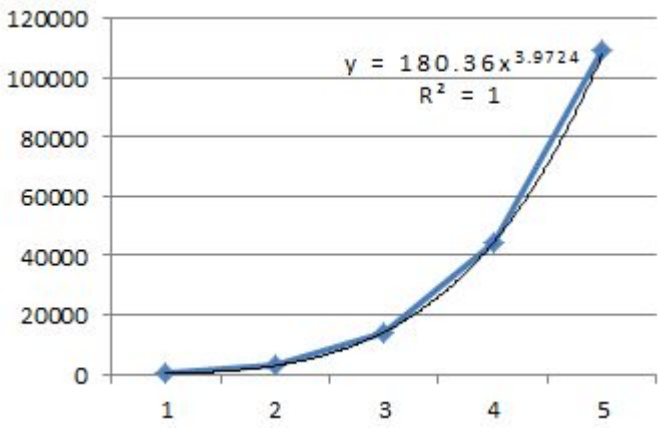

Chart 3 – Heyer QTC (DdN) Algorithm's Trendline

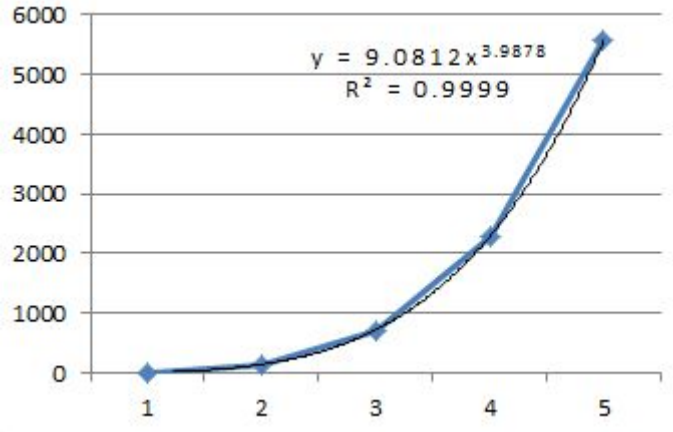

Chart 4 – Danalis QTC (DdP) Algorithm's Trendline

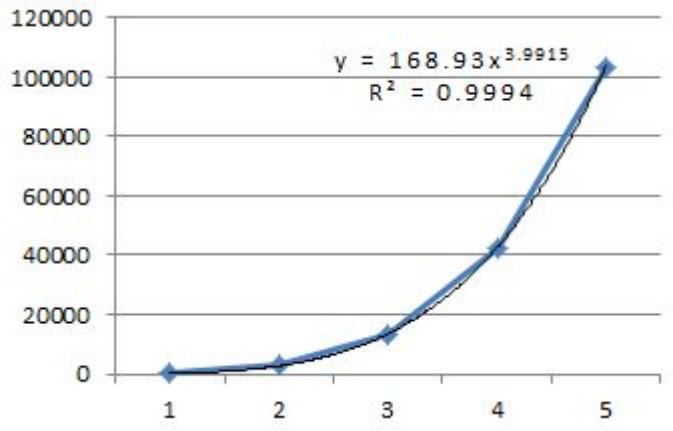

Chart 5 – Stochastic QTC (SdN) Algorithm's Trendline

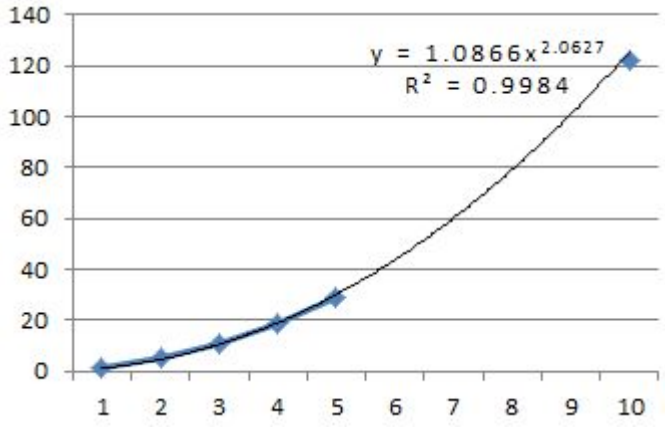

Chart 6 – QTC 1 (DrN) and QTC 2 (SrN) Algorithms' Trendline

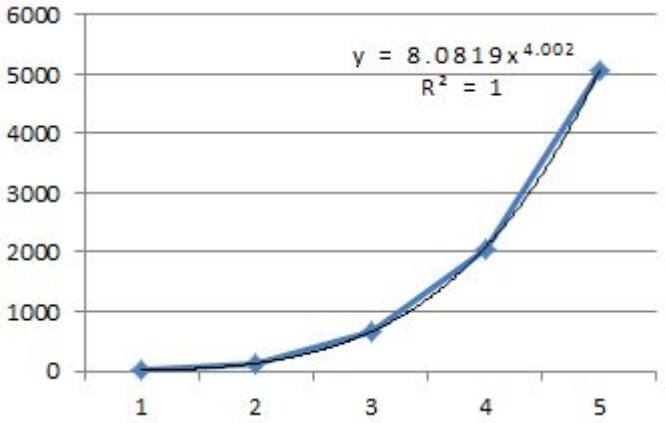

Chart 7 – QTC 3 (SdP) Algorithm's Trendline

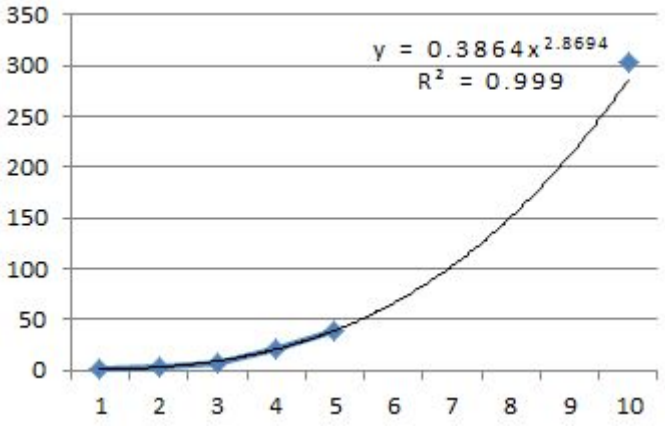

Chart 8 – QTC 4 (DrP) Algorithm's Trendline

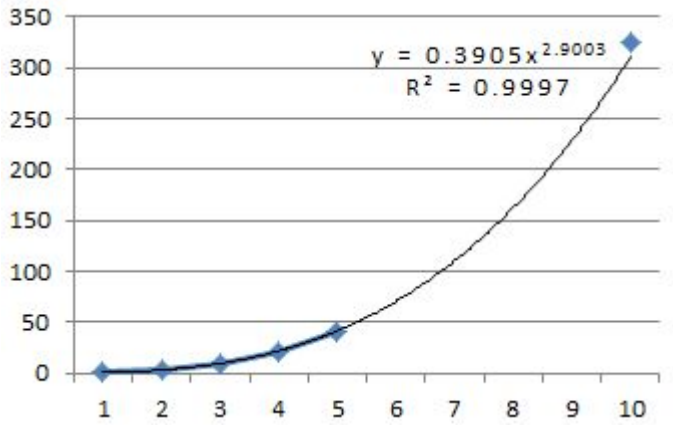

Chart 9 – QTC 5 (SrP) Algorithm's Trendline

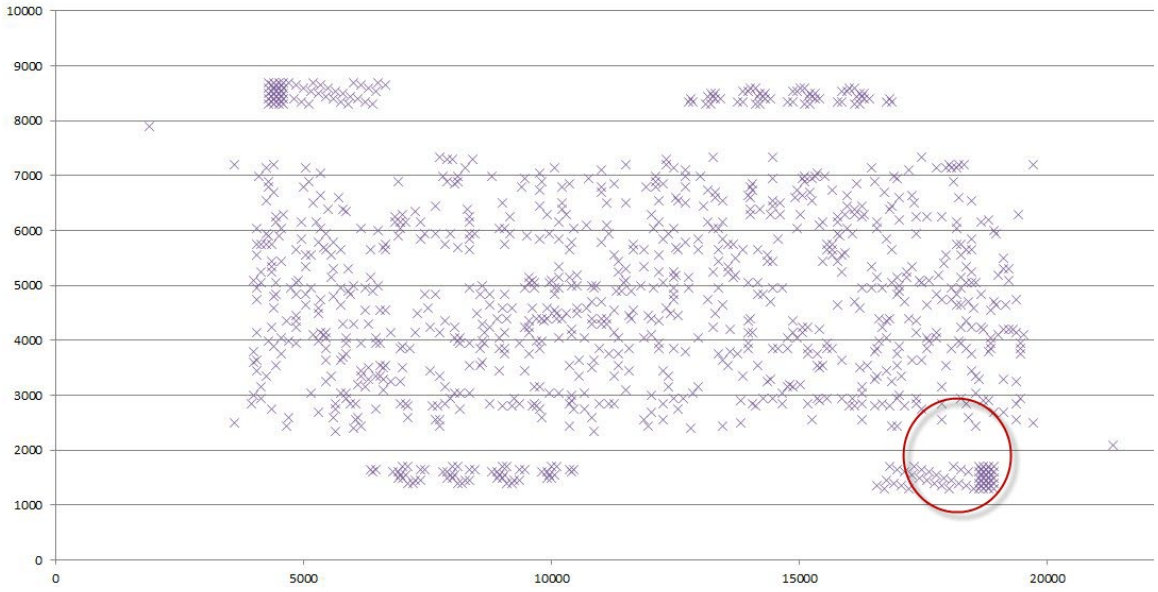

Chart 10 – Scatterplot of Dataset 1 with Cluster 1 Identified

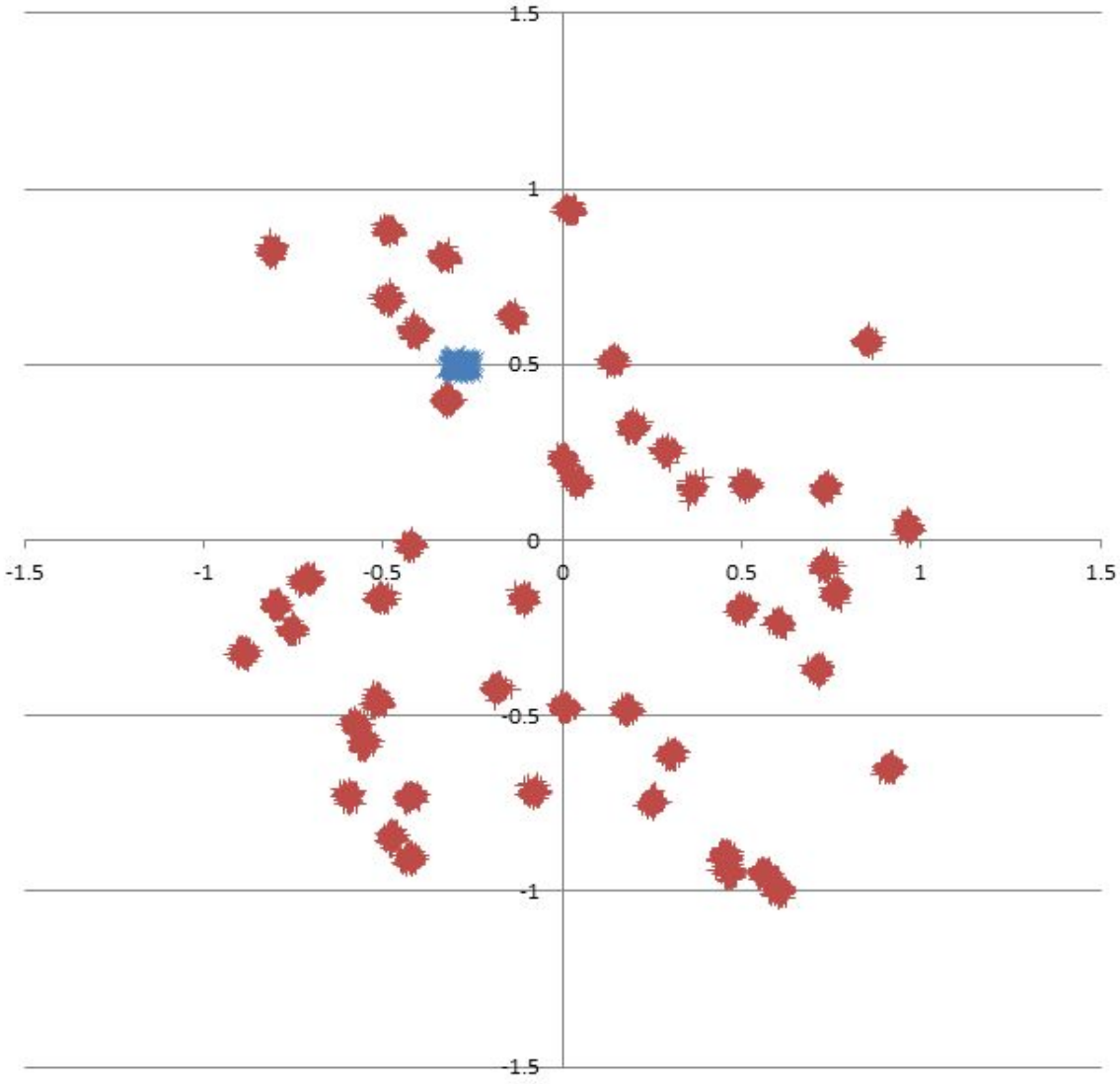

Chart 11 – Scatterplot of Dataset 2 with Cluster 1 Identified

References

Arthur, D., & Vassilvitskii, S. (2007, January). k-means++: The advantages of careful seeding. *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '07).*

Bednarik, L. & Kovacs, L. (2011). Extension of HAC clustering method with quality threshold. *IEEE 9th International Symposium on Intelligent Systems and Informatics (SISY 2011), Subotica, Serbia,* 257-261.

Bergholz, T., Wick, L., Qi, W., Riordan, J., Ouellette, L., & Whittman, T. (2007). Global transcriptional response of Escherichia coli O157: H7 to growth transitions in glucose minimal medium. *BMC Microbiology, 7*(97), 1-27.

Choudhury, N., Sarmah, R., & Sarma, S. (2012). *A modified QT-clustering algorithm over Gene Expression data.* Paper presented at the 2012 1st International Conference on Recent Advances in Information Technology (RAIT 2012), Dhanbad, India.

Coppe, A., Ferrari, F., Bisognin, A., Danieli, G., Ferrari, S., Bicciato, S., & Bortoluzzi, S. (2009). Motif discovery in promoters of genes co-localized and co-expressed during myeloid cells differentiation. *Nucleic Acids Research, 37*(2), 533-549.

Croce, D., Giannone, C., Annesi, P., & Basili, R. (2010). *Towards open-domain semantic role labeling. Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics.* Uppsala, Sweden.

Dan, O., & Mocian, H. (2009). *Scalable web mining with newistic.* Paper presented at the 13th Pacific-Asia Conference on Knowledge and Data Mining, Bangkok, Thailand.

Danalis, A., McCurdy, C., & Vetter, J. S. (2012). Efficient Quality Threshold Clustering for Parallel Architectures. *Proceedings of the 26th International Annual Symposium on Parallel and Distributed Processing,* IPDPS '12, 1068-1079.

Danielson, M.L., & Lill, M. A. (2010). New computational method for prediction of interacting protein loop regions. *Proteins-Structure Function and Bioinformatics, 78*(7), 1748-1759.

Duan, S., Fokoue, A., Srinivas, K., & Byrne, B. (2011). A Clustering-Based Approach to Ontology Alignment. *Proceedings of the 10th International Semantic Web Conference (ISWC 2011), Bonn, Germany.*

Dutta, S., & Overbye, T.J. (2011). A Clustering based Wind Farm Collector System Cable Layout Design. *Power and Energy Conference at Illinois (PECI),* 1-6.

Ferecatu, M., & Geman, D. (2009). A statistical framework for image category search from a mental picture. *IEEE Transactions on PAMI, 31*(4), 1087-1101.

Geremek, M., Bruinenberg, M., Ziętkiewicz, E., Pogorzelski, A., Witt, M., & Wijmenga, C. (2011). Gene expression studies in cells from primary ciliary dyskinesia patients identify 208 potential ciliary genes. *Human Genetics, 129*(3), 283-293.

Gu, Y., & Wang, C. (2010). *A study of hierarchical correlation clustering for scientific volume data.* Paper presented at the 6th International Symposium on Visual Computing, Las Vegas, NV.

Hara, H., Ohnishi, Y., & Horinouchi, S. (2009). DNA microarray analysis of global gene regulation by A-factor in Streptomyces griseus. *Microbiology-SGM, 155,* 2197-2210.

Ha-Thuc, V., Nguyen, D.C., & Srinivasan, P. (2008). A Quality-Threshold Data Summarization Algorithm. *IEEE International Conference on Research, Innovation and Vision for the Future RIVF, 2008,* 240-246.

Hevner, A.R., March, S.T., Park, J., & Ram, S. (2004). Design science in Information Systems research. *MIS Quarterly, 28*(1), 75-105.

Heyer, L. J., Kruglyak, S., & Yooseph, S. (1999). Exploring expression data: Identification and analysis of coexpressed genes. *Genome Research, 9*(11), 1106-1115.

Jiang, D., Pei, J., & Zhang, A. (2005). An interactive approach to mining gene expression data. *IEEE Transactions on Knowledge and Data Engineering, 17*(10), 1363-1378.

Laszlo, M., & Mukherjee, S. (2006). A genetic algorithm using hyper-quadtrees for low-dimensional K-means clustering. *IEEE Transactions on PAMI, 28*(4), 533-543.

Leisch, F. (2006). A toolbox for K-centroids cluster analysis. *Computational Statistics & Data Analysis, 51(2)*, 526-544.

Lesieutre, B., Rogers, K. M., Overbye, T. J., & Borden, A. (2010). Asensitivity approach to detection of market power potential. *IEEE Transactions on Power Systems, 26*(4), 1980-1988.

Minami, K., Maniratanachote, R., Katoh, M., Nakajima, M., & Yokoi, T. (2006). Simultaneous measurement of gene expression for hepatotoxicity in thioacetamide-administered rats by DNA microarrays. *Mutation Research-Genetic Toxicology and Environmental Mutagenesis, 603*(1), 64-73.

Minami, K., Saito, T., Narahara, M., Tomita, H., Kato, H., Sugiyama, H., . . . Yokoi, T. (2005). Relationship between hepatic gene expression profiles and hepatotoxicity in five typical hepatotoxicant-administered rats. *Toxicological Sciences, 87*(1), 296-305.

Nieto, X. G. (2010). Quality threshold clustering to determine the amount of parts in an object. Retrieved from http://bitsearch.blogspot.com/2010/09/quality-threshold-clustering-to.html

Olson, M., Epstein, J., Sackett, D., & Yergey, A. (2011). Production of reliable MALDI spectra with quality threshold clustering of replicates. *Journal of the American Society for Mass Spectrometry, 22*(6), 969-975.

Pawlik, A., Alibert, O., Baulande, S., Vaigot, P., & Tronik-Le Roux, D. (2011). Transcriptome characterization uncovers the molecular response of hematopoietic cells to ionizing radiation. *Radiation Research, 175*(1), 66-82.

Pukáncsik, M., Békési, A., Klement, E., Hunyadi-Gulyás, E., Medzihradszky, K., Kosinski, J., . . . Vértessy, B. (2010). Physiological truncation and domain organization of a novel uracil-DNA-degrading factor. *FEBS Journal, 277*(5), 1245-1259.

Reilly, C., Wang, C.C., & Rutherford, M. (2005). A rapid method for the comparison of cluster analyses. *Statistica Sinica, 15(1)*, 19-33.

Saito, N., Hatori, T., Aoki, K., Hayashi, M., Hirata, Y., Sato, K., . . . Iwabuchi, S. (2009). Dynamics of global gene expression changes during brain metastasis formation. *Neuropathology, 29*(4), 389-397.

Schafer, A., & Fey, D. (2008). Pollarder: An architecture concept for self-adapting parallel applications in computational science. *Proceedings of the 8th International Conference on Computational Science (ICCS 2008, Part 1)*. Kraków, Poland.

Scharl, T., & Leisch, F. (2006). The stochastic QT–Clust algorithm: evaluation of stability and variance on time–course microarray data. *Proceedings in Computational Statistics.* Heidelberg, Germany.

Scharl, T., & Leisch, F. (2010). *Quality-Based Clustering of Functional Data: Applications to Time Course Microarray Data.* Paper presented at the 32nd Annual Conference of the German-Classification-Society, Helmut Schmidt Univ, Hamburg, Germany.

Scharl, T., Striedner, G., Pötschacher, F., Leisch, F., & Bayer, K. (2009). Interactive visualization of clusters in microarray data: An efficient tool for improved metabolic analysis of E. coli. *Microbial Cell Factories, 8*(37), 1-12.

Sidorov, K., Hicks, Y., Marshall, D., Sanei, S., & Chambers, J. (2006). *Real time multi camera 3D tracking system.* Paper presented at the 3rd European Conference on Visual Media Production (CVMP 2006), London, UK.

Srinivasan, V., Stankovic, J., & Whitehouse, K. (2013). FixtureFinder: discovering the exitence of electrical and water fixtures. *Proceedings of the 12th international conference of Inforamtion processing in sensor networks (IPSN '13).* Philadelphia, PA, USA.

Tanaka-Tsuno, F., Mizukami-Murata, S., Murata, Y., Nakamura, T., Ando, A., Takagi, H., & Shima, J. (2007). Functional genomics of commercial baker's yeasts that have different abilities for sugar utilization and high-sucrose tolerance under different sugar conditions. *YEAST, 24(10)*, 901-911.

Tang, Z., Zhang, L., Cheema, A.K., & Ressom, H.W. (2010). A New Method for Alignment of LC-MALDI-TOF Data. *2010 IEEE International Conference on Bioinformatics and Biomedicine.* 346-351

Toledo-Rodriguez, M., Goodman, P., Illic, M., Wu, C., & Markram, H. (2005). Neuropeptide and calcium-binding protein gene expression profiles predict neuronal anatomical type in the juvenile rat. *Journal of Physiology-London, 567*(2), 401-413.

Waoo, N., Kashyap, R., & Jaiswal, A. (2010). DNA Nano Array Analysis Using Hierarchical Quality Threshold Clustering.

Yaakob, S.N., & Jain, L. (2012). An insect classification analysis based on shape features using quality threshold ARTMAP and moment invariant. *Applied Intelligence, 37(1)*, 12-30.

Yaakob, S. N., Lim, C. P., & Jain, L. (2009). *A novel euclidean quality threshold ARTMAP network and its application to pattern classification.* Paper presented at the 12th International Conference on Knowledge-Based Intelligent Information and Engineering Systems, Zagreb, Croatia.

Yager, R. R. (2008). Some measures relating partitions useful for computational intelligence. *International Journal of Computational Intelligence Systems, 1*(1), 1-18.

Yuan, Z., Tan, R., Xing, G., Lu, C., Chen, Y., & Wang, J. (2008). *Fast sensor placement algorithms for fusion-based target detection.* Paper presented at the Real-Time System Symposium 2008 (RTSS), Barcelona, Spain.

Zhang, T., Ramakrishnan, R., & Livny, M. (1996). BIRCH: An efficient data clustering method for very large databases. *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data (SIGMOD '96).*