MENU OPTIMIZATION WITH LARGE-SCALE DATA

JEYHUN KARIMOV

MASTER THESIS

THE DEPARTMENT OF COMPUTER ENGINEERING

TOBB UNIVERSITY OF ECONOMICS AND TECHNOLOGY

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

DECEMBER 2015

ANKARA

Approval of the Graduate School of Natural and Applied Sciences.

Prof. Dr. Osman EROĞUL
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Murat Alanyalı
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Ahmet Murat Özbayoglu
Supervisor

Examining Committee Members

Chair : Prof. Dr. Ali Aydın SELÇUK

Member : Asst. Prof. Dr. Ahmet Murat Özbayoglu

Member : Prof. Dr. Mehmet Önder Efe

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada orijinal olmayan her türlü kaynağa eksiksiz atıf yapıldığını bildiririm.

I hereby declare that all the information provided in this thesis was obtained with rules of ethical and academic conduct. I also declare that I have sited all sources used in this document, which is written according to the thesis format of the Institute.

Jeyhun KARIMOV

University	**:**	**TOBB University of Economics and Technology**
Institute	**:**	**Institute of Natural and Applied Sciences**
Science Programme	**:**	**Computer Engineering**
Supervisor	**:**	**Asst. Prof. Dr. Ahmet Murat Özbayoglu**
Degree Awarded and Date	**:**	**M.Sc. – December 2015**

Jeyhun KARIMOV

MENU OPTIMIZATION WITH LARGE-SCALE DATA

ABSTRACT

The use of optimal menu structuring for different customer profiles is essential because of usability, efficiency, and customer satisfaction. Especially in competitive industries such as banking, having optimal graphical user interface (GUI) is a must. Determining the optimal menu structure is generally accomplished through manual adjustment of the menu elements. However, such an approach is inherently flawed due to the overwhelming size of the optimization variables' search space. We propose a solution consisting of two phases: grouping users and finding optimal menus for groups. In first part, we used H$(EC)^2$ S , novel Hybrid Evolutionary Clustering with Empty Clustering Solution. For second part we used Mixed Integer Programming (MIP) framework to calculate optimal menu. We evaluated the performance gains on a dataset of actual ATM usage logs. The results show that the proposed optimization approach provides significant reduction in the average transaction completion time and the overall click count.

Keywords: Optimization, Bigdata, clustering,Automated Teller Machine.

Üniversitesi : **TOBB Ekonomi ve Teknoloji Üniversitesi**
Enstitüsü : **Fen Bilimleri**
Anabilim Dalı : **Bilgisayar Mühendisliği**
Tez Danışmanı : **Asst. Prof. Dr. Ahmet Murat Özbayoglu**
Tez Türü ve Tarihi : **Yüksek Lisans – December 2015**

Jeyhun KARIMOV

BÜYÜK VERİ İLE MENÜ ENİYİLEMESİ

ÖZET

Farklı müşteri profilleri için en uygun menü kullanımı kullanılabilirlik, verimlilik ve müşteri memnuniyeti açısından esastır. Özellikle bankacılık gibi rekabetçi sektörlerde, en iyi menü kullanıcı arayüzüne sahip olmak bir zorunluluktur. Optimal menü yapısının belirlenmesi genellikle menü elemanının manuel ayarlanması ile gerçekleştirilir. Ancak, bu metot özellikle kompleks menülerde işe yaramaz. Bu çalışmada iki aşamadan oluşan çözüm önerilmiştir: kullanıcıları gruplandırmak ve gruplar için en uygun menüler bulmak. İlk bölüm için H$(EC)^2$S, yeni hibrid Evrimsel Kümeleme algoritmasını geliştirdik. Ikinci bölümde optimal menü hesaplamak için Karışık Tamsayılı Programlama kullandık. Sonuçları gerçek ATM logları üzerinde test ettik ve performans artımı olduğunu gözlemledik.

Anahtar Kelimeler: Eniyileme, büyük veri, kümeleme,bankamatik.

ACKNOWLEDGEMENTS

I would like to thank my supervisor Asst. Prof. Dr. Ahmet Murat Özbayoglu, who gave me enormous, comprehensive support and high motivation to complete this thesis, conduct the complex research consisting of the implementation part, test, user study, survey and statistical analysis necessary for the thesis and a number of other research through my master degree period.

I thank Prof. Dr. Erdoğan Doğdu, Prof. Dr. Bulent Tavlı and Asst. Prof. Dr. Buğra Caşkurlu for their contributions to my research.

My gratitude to TOBB University of Economics and Technology for providing me a scholarship throughout my study and to all staff members and assistants of the Computer Engineering Department of TOBB University of Economics and Technology for their deep teaching, advice, suggestions, which I appreciate the most.

I also thank Turkish Ministry of Science and Technology for their support provided as part of project No. 0.367.STZ.2013-2 and Provus Inc for providing me the data I used during my work.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Graphical User Interface (GUI) is the key point in multi-profile customer systems. Multi-profile customer systems can be considered as the systems that include many customers of different types. For example, web site, Automatic Teller Machine (ATM) or some phone app users can be divided to several types in terms of their behaviors or actions. The efficiency of menus in GUI can be calculated with click count, overall time, average time per transaction, and etc. to finish the required action. Are the existing menu structures are designed according to these requirements? Do they guarantee the optimum? These questions are important for both customers and industry. For example, customers can wait less time in ATM queues and companies can get customer satisfaction in return.

From initial perspective, menu optimization can be seen as a trivial problem. For example, finding the most clicked menu items and putting them to initial menu screens, is one solution. Moreover, when looking to menu items and their overall click counts, one can redesign the menu in a more optimized way. However, as menu structure gets complex we need deterministic solution, away from heuristics, guaranteeing the optimum. One important thing is that, any false assumption in heuristical based solutions can cost to companies to lose their customers.

Many researchers considered this problem to make better menus in GUI. Some solutions considered designing menu for people with disabilities [2, 3]. Yet in another solution ATM menus were designed based on questionnaires on customers [4]. All previous research we investigated, propose a menu model for GUI, based on some pre-assumptions. On the other hand, to be usable, the solution for optimal menu problem should be generic and applicable. For example, the solution should have no information about user types, their knowledge about system, the location of the system, questionnaires and etc. Moreover, the expected solution should be amorphous according to its application area and easily adjustable. For example, some company may need a menu with minimum click counts, other with minimum time, yet another with maximum menu item

size and visibility. Furthermore, usability is one of the main factors affecting the quality of software systems [5, 6, 7]. Therefore, the optimal menu has to ensure its usability. For example, the structure of menu cannot contain any disambiguous placement of menu items. So, the optimal menu is not just one solution but generic framework.

To design such a menu, we propose a 2-stage solution: clustering users and finding optimum solution for each group. For the first part, we developed a combined solution which encompasses fast and quality clustering within it. For the second part, mixed integer programming (MIP) framework is proposed.

The structure of this study is as follows. After this short introduction, we give literature review in Section 2. Then we provide a detailed explanation and solution of the problem in Section 3. Experiments are provided in Section 4. Possible applications of our model to various research areas and open research problems are given in Section 5. Our future work based on this method and conclusions are provided in Section 6.

2. RELATED WORK

There are various research conducted on clustering and GUI optimization. We grouped related works into categories.

2.1 Clustering

Because of its simplicity and applicability k-means [8] is the most widely used algorithm to cluster data. There are some studies implemented on optimizing different objectives of k-means algorithm such as Euclidean k-medians [9, 10] and geometric k-center [11]. Minimization of the sum of distances to the nearest center is the goal for Euclidean k-medians, and minimization of the maximum distance from every point to its nearest center is the one for geometric k-center version. Another research was done to seek a better objective function of k-means [12]. Although there are different versions of k-means that might have advantages, parallelization o the algorithm in a single machine resulted in significant performance improvement [13, 14]. Achieving the parallelization over multiple machines results in even better improvements. The MapReduce framework [1, 15] provides significant improvements to scalable algorithms.

There has been several studies for clustering large scale data on distributed systems in parallel on Hadoop [16]. One such approach is Haloop [17], which is a modified version of the Hadoop making the task scheduler loop-aware and by adding various caching mechanisms. Another approach to cluster data in a distributed system was using Apache Mahout library [18]. Moreover, clustering of big data can be done on cloud also. In [19], the tests were running on Amazon EC2 instances and the comparisons were made to realize the gain between the nodes. Esteves et al. made comparisons over k-means and fuzzy c-means for clustering Wikipedia large scale data set [20]. Both in [20] and [19], the authors used Apache Mahout for clustering data.

2.2 Hierarchical Menu Optimization

Hierarchical menu optimization has been the main topic of many research works. In [21], authors developed an analytical model for search time in menus. The advantage of menu's breath over its depth is stated in [22]. Researchers also showed experimentally that using broader and shallower menus instead of deeper and narrower ones make it easier and faster to access information [23, 24]. There are also hybrid solutions; in [25, 26] it is found that menus with larger breadth at deeper layers were more efficient than menus that became narrower towards the end. One of the earliest research conducted on menu optimization, [27], used the hierarchical index of a digital phone book using the access frequencies of phone numbers. Another approach is called "split menus"[28] where frequently accessed menu items are located at the top of the menu groups or menu pages by splitting the menus. In [29], quantitive approach for hierarchical menu design was proposed by authors. Using Huffman Code to optimize the menu structure based on the probabilities of menu items' access times is another approach that was proposed in [30].

2.3 Heuristic Solutions

There are some recent works that focus on using evolutionary algorithms and heuristics for menu optimization. Guided-Search algorithm is proposed in [31] for defining the necessary components of a good user interface. The authors of [32] used and tested the Guided-Search algorithm by analyzing the algorithm's performance under different scenarios in order to increase the satisfaction level. Genetic algorithms were used for menu structure and color scheme selection in [33, 34] and in [35] authors used genetic algorithm with simulated annealing for optimizing the performance of menus on cell phones. Automated menu optimization and design is desirable but semantics always play a role. Therefore, some researchers suggested using human assistance as part of the optimization

and design process. In [36] authors present MenuOptimizer, a menu design tool that employs interactive optimisation approach to menu optimisation. The tool allows designers to choose good solutions and group items while delegating computational problems to an ant colony optimizer. [37] also suggests using *informal judgements* for the final menu structure along with an automated procedure.

Authors discuss GUI optimization techniques for evaluating and improving cell phone usability with efficient hierarchical menu design in [38]. Another research about cell phone menu optimizations, utilized menu option usage frequencies and recent usage history to find the best menu option ranking by using Ranking SVM method [39]. ClickSmart system was proposed in [40], to adapt WAP menus to mobile phones.

Fireworks algorithm, which was inspired by observing fireworks explosion, is a recent meta-heuristic that was proposed by Tan et al [41]. Authors showed that it outperforms Standard PSO and Clonal PSO in experiments. Enhanced Fireworks Algorithm is the improved version of the Fireworks algorithm [42], which we used in our model. Cuckoo search is another meta-heuristic that was proposed recently by Yang et al. [42], which was inspired by obligate brood parasitic behavior of some cuckoo species in combination with the Levy flight behavior of some birds and fruit flies.

2.4 ATM Menu Optimization

There have been several studies to optimize the menu structure on ATM [43, 44, 45, 46, 47] to reduce click counts and therefore waiting queues. The main objective is to display a menu that is optimal or results in less click counts to finish the required tasks for all users. Online banking user interface optimized with the same objective in [48]. Web site menu optimization is done with hill-climbing method which minimizes average time to reach target pages in [49].

Some works on GUI optimization make specific assumptions about users before optimization and optimize user menus accordingly. Authors emphasize the contributions of cognitive psychology and human factors research in the design of menu selection systems in [22]. In [50], authors concentrated on optimizing ATM menus for the usage of student communities. In [51] on the other hand, authors optimized ATM menus for illiterate people and tested on a group of six Dutch and functionally illiterate persons. Mobile interface GUI optimization was done for novice and low-literate customer usages in [52]. Designing ATM menus with speech-based and an icon-based interfaces for literate and semi-literate groups was conducted in [53]. In [54], authors reviewed a lot of different studies performed on Human Computer Interaction Technology for ATMs where they included different ATM design and implementations.

To the best of our knowledge, our solution is the first work on menu optimization which guarantees the optimum solution. In our generic GUI optimization framework in general and ATM menu optimization in particular, our solution is independent of any human assumptions, heuristics and other factors.

3. PROPOSED SYSTEM

We propose a solution which consists of two parts. To make better user menus it is crucial to cluster similar users and make menus for each cluster. In this way it is possible to get better performance and quality.

3.1 Clustering Part

3.1.1 MapReduce Framework

MapReduce is a programming model, mainly inspired by functional programming, used to process and generate large datasets with parallel and distributed algorithm on a cluster[1, 15]. MapReduce is useful in a wide range of applications, including distributed pattern-based searching, distributed sorting, web link-graph reversal, document clustering, machine learning[55]. MapReduce's stable inputs and outputs are usually stored in a distributed file system. The transient data is usually stored on the local disk and fetched remotely by the reducers. The computational processing can occur on both unstructured and structured data. The programming model is as follows [1]:

- The computation takes a set of input key/value pairs, and produces a set of output key/value pairs.The user of the MapReduce library expresses the computation as two functions: Map and Reduce.
- Map, written by the user, takes an input pair and produces a set of intermediate key/value pairs: Map(k1,v1) $\rightarrow$ list(k2,v2).The MapReduce library groups together all intermediate values associated with the same intermediate key and passes them to the Reduce function.
- The Reduce function, also written by the user, accepts an intermediate key and a set of values for that key: Reduce(k2, list (v2)) $\rightarrow$ list(v3).

It merges together these values to form a possibly smaller set of values. Typically just zero or one output value is produced per Reduce invocation. The intermediate values are supplied to the user's reduce function via an iterator. This allows us to handle the lists of values that are too large to fit in memory.

Figure 3.1 illustrates the MapReduce model Here the user program forks

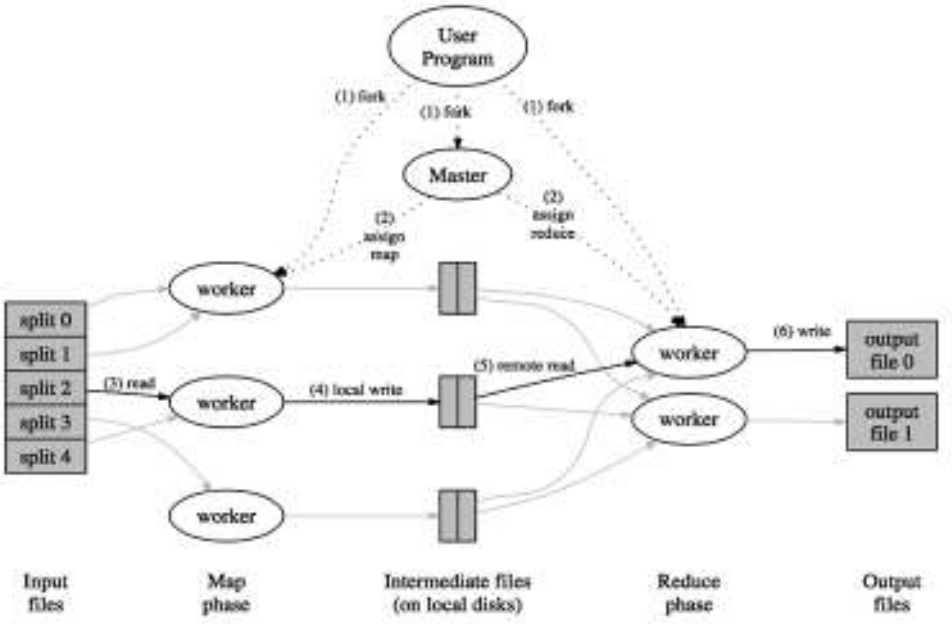


Figure 3.1: Execution overview of MapReduce

the process to all nodes. Then the master node assigns the roles and the responsibilities of mappers and reducers. After that, the input is splited into several partitions to be processed in parallel for mappers-workers. After all of the data is read, the map phase is finished. The input of the reducers is located on the local disks. Then the reduce phase begins and each reducer produces their output. More compact view of this model is shown in Figure 3.2

There has been several studies for clustering large scale data on distributed systems in parallel on Hadoop[16]. One such approach is Haloop[17], which is a modified version of the Hadoop MapReduce framework. The proposed model

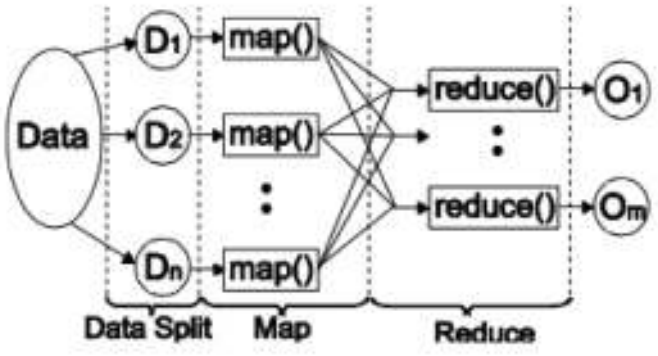


Figure 3.2: Compact MapReduce programming model

dramatically improves the efficiency by making the task scheduler loop-aware and by adding various caching mechanisms. Authors used the k-means algorithm to evaluate their model against the traditional one and as a result, the proposed model reduced the query runtimes by 1.85.

Another approach to cluster data in a distributed system was using Apache Mahout library. Research was done to cluster data in the cloud [19]. The tests were running on Amazon EC2 instances and the comparisons were made to realize the gain between the node numbers. Yet another study was done to cluster Wikipedia's latest articles with k-means [20].

Another research was concentrated on MapReduce model's not directly supporting processing multiple related heterogeneous datasets [56]. Authors called their model Map-Reduce-Merge. It adds a Merge phase to the standard model. This phase can efficiently merge the data already partitioned and sorted by the map and reduce modules.

Meanwhile, working with large data sets in parallel and clustering them efficiently requires scalable and multi node computing machines. MapReduce architecture [1] is one particular example of such a system which had been in use more often lately. There are advantages of MapReduce over parallel databases like storage-system independence and fine-grain fault tolerance for large jobs[57].Because

MapReduce model works on multicore systems, there are some research done to evaluates the suitability of the this model for multi-core and multi-processor systems[58]. Authors of this research study Phoenix with multi-core and symmetric multiprocessor systems. Afterwards, they evaluate its performance potential and error recovery features. Moreover, they also compare the codes of MapReduce and P-threads which is written in lower-level API. As a result, authors conclude that MapReduce is a promising model for scalable performance on shared-memory systems with simple parallel code. MapReduce model is mostly used in offline jobs, because of its processing large data and late response time. However, authors of [59] researched the online version of Hadoop MapReduce framework.They propose a solution that allows users to see 'early returns' from a job while it is being computed and process continuous queries on the framework.

3.1.2 k-means algorithm

k-means is one of the simplest unsupervised learning algorithms that attempts to solve the well known clustering problem[8]. In k-means clustering, we are given a set of n data points in d-dimensional space and an integer k and the problem is to determine a set of k points within n data points, called centers or centroids, so as to minimize the sum of mean squared distance from each data point to its nearest center. Initially, k centroids must be defined, one for each cluster. These centroids should be placed carefully because different location choices can result in different cluster formations. That is why, some heuristics suggest to place the centroids far away from each other as much as possible [60]. Afterwards, for each point that belongs to the given data set, it is associated with the nearest centroid. Then the new centroid is recalculated from the data points associated with each centroid. After calculating k new centroids, the algorithm must be iterated once again to find and associate each point to its nearest centroid. This loop continues as long as the locations of the new centroids keep changing until they all remain unchanged. The aim is to minimize the objective function given in (3.1)

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^j - c_j \right\|^2 \tag{3.1}$$

where $\left\| x_i^j - c_j \right\|$ is a distance measure between a data point x_i^j which is i^{th} element of the data set n belonging to j^{th} centroid ,and the cluster center c_j. To wrap up, the pseudo-code for the algorithm can be rewritten in this way:

1. Select k points in the space of objects being clustered. These points are initial centroids.

2. Calculate each object's distance against all of the centroids and assign each object to the group that has the closest centroid

3. When all objects have been assigned, recalculate the positions of the k centroids by the formula:

$$c_{j+1} = \frac{1}{|S_j|} \sum_{i=1}^{n} x_i \tag{3.2}$$

   where c is the new centroid,S_j is the number of x objects associated with the j^{th} centroid

4. If any of the new centroids that was found in step 3 is different from the previous one, repeat Steps 2 and 3, else exit.

Time complexity of the k-means algorithm is $O(mn)$ where n is the number of data points and m is the number of iterations. The number of iterations m tends to increase when the number of clusters (k) increases, however the number of data points that change their associated centroids tend to decrease during the consequent iterations, however the original k-means algorithm does not utilize this phenomena, i.e. the number of operations performed at every iteration is the same regardless of the number of data points that actually change their centroids. Authors used MapReduce framework to implement k-means in

parallel environment also[61]. One of the models we used to evaluate and compare proposed solutions was [61].

In this first part of the study, we propose a simple heuristic that is aimed to utilize the decreasing number of operations in these further iterations for k-means. We implemented the algorithm both serially and in parallel to evaluate how much efficiency is obtained under different scenarios.

3.1.3 Fireworks Algorithm

Enhanced Fireworks algorithm (EFWA) is a swarm intelligence algorithm and adopts four parts to solve optimization problems. The first part is the explosion operator. Suppose the population of EFWA consists of N D-dimensional vectors as individuals, each individual 'explodes' according to the explosion operator and generates sparks around it. The second part is the mutation operator. Sparks are generated under the effect of this operator in order to improve the diversity of the population. The third part, namely as the mapping rule, is used to map the sparks that are out of the feasible space into it. By applying this rule, the sparks are pulled back to feasible space. The last part is called as selection strategy. The individuals are selected from the whole population and passed down to the next generation.

When a firework is set off, a shower of sparks will fill the local space around the firework. In our opinion, the explosion process of a firework can be viewed as a search in the local space around a specific point where the firework is set off through the sparks generated in the explosion. Mimicking the process of setting fireworks, a rough framework of the algorithm is depicted in Figure 3.3.

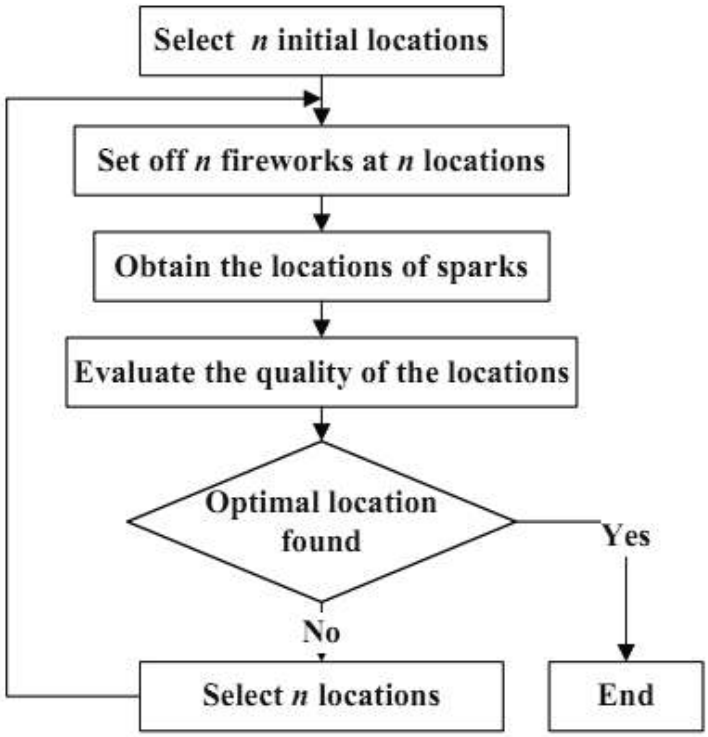


Figure 3.3: Intuition behind EFWA

3.1.4 Clustering Performance Improvement

k-means is the best known algorithm for clustering for its usability and scalability. Although there are numerous modifications of the k-means algorithm, both in single machine and in MapReduce model, the complexity of the algorithm more or less remains the same. In particular, Standard k-means (k-means-s) model can not escape from the complexity of the algorithm where the new centroids and nearest centroids are recalculated each time [61]. So, in each iteration:

- First part (P1) - All points are processed and the nearest centroids are found
- Second part (P2) - All points are processed in k groups to find the new centroids

As can be seen above, reprocessing of all points twice in each iteration increases the computation time of the algorithm linearly as the data and k value gets bigger. Improvement for the first part was done in [62]. However, this improvement have some disadvantages when working with big data. That is why, we further improved the model proposed in [62] and proposed two new solutions for the parallel computation with big data and the serial computation with non-big data. We will show the threshold between big data and non-big data for the data sets we used, in Section 4.

The models we propose, namely k-means-inbd (k-means improved for non-big data or k-means Improved for Serial Computation) and k-means-ibd (k-means improved for big data or k-means Improved for Parallel Computation), eliminates the majority of the complexity associated with both parts of the standard parallel k-means [61]. This improvement decreases the computation time and the complexity of the algorithm considerably. So, let

- x_i denote a single data point and S denote all set of points in data set,where $x_i \in S \forall i$.
- $c_j^t \in C^t$ denote the centroid computed in t^{th} iteration, where $j = \{1, 2, 3, ..., k\}$
- S_j^t denote the set of data points belonging to c_j^t
- P_j^t denote the set of newly accepted points to cluster representing c_j^t
- M_j^t denote the set of outgoing points from cluster representing c_j^t
- z_i denote the distance between x_i and its belonging centroid.
- v_i denote the index of x_i^{th} belonging centroid, c_k^t from the set C^t
- α be a constant value denoting threshold value where $0 < \alpha < 1$.

The first proposed model is k-means-inbd. General procedure of this model is as follows:

In the first iteration of k-means-inbd, for all data points, their nearest centroids are calculated and for each $x_i \in S$, we keep z_i and v_i. After that, new centroids are calculated as in k-means-s. Beginning from $t = 2^{th}$ iteration, when computing the nearest centroids for each data point, $x_i \in S$, we calculate $d^t(x_i\ ,c^t_{v_i})$, distance between current data point and its previous centroid's new value. That is, $c^t_{v_i}$ is the newly computed value of centroid $c^{t-1}_{v_i}$. If $d^t \leq z_i$, then x_i stays in same cluster. Thus, it can be ignored during the recalculation of the new centroid. Otherwise, it means that x_i has changed its cluster and it must be considered while recalculating the new centroids. After processing all data points, only those that were chosen are considered in the calculation of the new centroids. So, the calculation of a new centroid is shown with Formula (3.3):

$$c^t_j = \frac{c^{t-1}_j * |S^{t-1}_j| - (\sum_{i=1}^{a} m^t_i) + (\sum_{i=1}^{b} p^t_i)}{|S^{t-1}_j| - a + b} \tag{3.3}$$

where $c^t_j \in C^t$ is the j^{th} centroid among k centroids at t^{th} iteration, $|S^{t-1}_j|$ is the number of points belonging to c^{t-1}_t, $m^t_i \in M^t_j$ is i^{th} point that is drawn out from j^{th} cluster at t^{th} iteration, $p^t_i \in P^t_j$ is i^{th} point that is added to j^{th} cluster at t^{th} iteration, $b = \left|P^t_j\right|$ and $a = \left|M^t_j\right|$. The pseudocode of k-means-inbd is shown in Algorithm (1).

The second proposed model is k-means-ibd (k-means improved for big data). The general structure of algorithm is as follows:

- Before threshold part (BT) - For i iterations until reaching threshold value, run as k-means-inbd. At each iteration t, compute threshold value α_t.
- After threshold part (AT) - if $\alpha_t > \alpha$, then run Algorithm (3).

The first i iterations until reaching threshold value is as the same as k-means-inbd. Beginning from $(i+1)^{th}$ iteration, we store centroids of previous iteration and size of all clusters. The iteration number i is decided as a result of threshold value α. That is, if the division of data points that changed their clusters to all

Algorithm 1

procedure k-MEANS-INBD($x_i \in S, \forall i$, k)
  Require: $S = \{x_0, x_1 ... x_n\}$, k is number of clusters
  Ensure: $c_1, c_2, ..., c_k$ centroids
  Initialize centroids for $t = 1$.
  Run k-means with its standard execution for the first iteration and keep z_i and v_i $\forall$ $x_i \in S$.
  Initialize C^t, set of resulting centroids at the end of iteration $t = 1$.
  while $c_j^t \neq c_j^{t-1}$ $\forall j$ **do**
    $t = t + 1$.
    for all $x_i \in S$ **do**
      Compute distance $d^t = d(x_i, c_{v_i}^t)$ and $d^{t-1} = z_i$.
      if $d^t \leq d^{t-1}$ **then** continue.
      else
        Compute $c_b^t \in C^t$, x_i^{th} new associated centroid from set C^t, where $b \neq j$.
        $P_b^t = P_b^t \bigcup x_i$, add x_i to set of new coming points for centroid c_b^t
        $M_j^t = M_j^t \bigcup x_i$, add x_i to the set of outgoing points for centroid c_j^{t-1}.
      Save x_i^{th} associated z_i and v_i, to use in the next iteration.
    Compute the new centroids using Formula (3.3).

data points in t^{th} iteration ($\alpha_t = \frac{|P^t|}{|S|}$) is less than predefined threshold (α), then we can be confident about clusters' being mostly stable. Important point is that, after threshold value is satisfied, in AT part, we do not keep all points' associating centroids, but the set of newly computed centroids, which can fit in memory in big data sets. In AT part, when recomputing the object assignments to the new centroids, the first to consider is the previous centroid. First we compute x_i^{th} previous nearest centroid, c_j^{t-1}. When computing the new centroid, begin from j^{th} centroid from C^t, namely c_j^t. If $d^{t-1}(x_i, c_j^{t-1}) \geq d^t(x_i, c_j^t)$, it means that the x_i stayed in the same cluster and there is no need to consider this data point when computing the new centroids. Otherwise, the data point has changed its cluster and it must be considered while recalculating a new centroid. Recalculating the new centroid part is the same as k-means-inbd. The pseudocode of k-means-ibd is shown in Algorithm (2).

Algorithm 2

1: **procedure** k-MEANS-IBD($x_i \in S, \forall i$, k)
   Require: $S = \{x_0, x_1 ... x_n\}$, k is number of clusters
   Ensure: $c_1, c_2, ..., c_k$ centroids
2: **while** $\alpha_t > \alpha$ **do**
3: Run Algorithm (1) for 1 iteration.
4: Compute $\alpha_t = \frac{|P^t|}{|S|}$, overall fraction of points that changed their existing clusters. Here $|P^t| = |M^t|$
5: Run Algorithm (3) with required parameters from this algorithm.

Since we compared both the parallel and the serial versions of the proposed models, MapReduce version of k-means-ibd and k-means-inbd was also implemented. The algorithm is the same. However, the mapping of the serial version's first part (evaluating each data point's nearest center) to the parallel version's mapper phase and the serial version's second part (calculating the new centroids after all data points chose their nearest centers) to the parallel version's reducer phase is enhanced. That is, in the mapper phase we find the data point's nearest centroid and in the reducer phase the new centroids are calculated from the points that changed their cluster.

3.1.4.1 Analysis of Proposed Models

When analyzing the proposed models, we can divide the k-means-inbd and k-means-ibd into two parts as k-means-s:

- First part - All points are processed and the nearest centroids are found.
- Second part - All points are processed in k groups to find the new centroids.

Both k-means-inbd and k-means-ibd have improvements in the second part. If we examine Figure 3.4 which explains Formula (3.3), it can be seen that there are two main subparts denoted as 1 and 2. First, denoted as 1, is a computationally

Algorithm 3

procedure k-MEANS-IBD-AT($x_i \in S, \forall i$, k, C^t)

Require: $S = \{x_0, x_1 ... x_n\}$, k is number of clusters, C^t set of current centroids.

Ensure: $c_1, c_2, ..., c_k$ centroids

Initialize centroids for $t = 1$.

while $c_j^t \neq c_j^{t-1}$ $\forall j$ where $0 < j \leq k$ **do**

$t = t + 1$.

for all $x_i \in S$ **do**

Among previous centroids, find the nearest centroid $c_j^{t-1} \in C^{t-1}$ to x_i and compute distance between them $d_j^{t-1} = d(x_i, c_j^{t-1})$.

Find $d_j^t = d(x_i, c_j^t)$, distance from x_i to j^{th} cluster from C^t, where j is index of c_j^{t-1}.

if $d_j^t \leq d_j^{t-1}$ **then** continue.

else

Compute $c_b^t \in C^t$, x_i^{th} new associated centroid from set C^t, where $b \neq j$.

$P_b^t = P_b^t \bigcup x_i$, add x_i to set of new coming points that belong to centroid c_b^t.

$M_j^t = M_j^t \bigcup x_i$, add x_i o the set of outgoing points that belonged to centroid c_j^{t-1}.

Save only the set C^t to be used in the next iteration.

Compute the new centroids using Formula (3.3).

constant time operation. As we will discuss in the experimental results, it is seen that second part, denoted as 2, consists of a small minority of the points of the whole data set, so as the iterations progress, the number of operations keep decreasing geometrically, i.e. the total number of operations converges to a constant; hence the whole formula can become a constant time operation.

$$c_j^t = \frac{c_j^{t-1} * |S_j^{t-1}| - (\sum_{i=1}^{a} m_i) + (\sum_{i=1}^{b} p_i)}{|S_j^{t-1}| - a + b}$$

Figure 3.4: Improvement done on second part of k-means algorithm

While analyzing the first part of proposed models, in k-means-ibd model, we are interested in minimizing the overall data sent from *mapper* to *reducer* phase and minimizing I/O time. The main advantage of k-means-ibd is that, it does not

change the original data after threshold is satisfied. So, there is no disc-write overhead, in any iteration after $\alpha_t < \alpha$. Important point here is that, after threshold value is satisfied, overall points tend to stay in their existing clusters, as we will see in Section 4. Therefore, we switch to Algorithm (3). This algorithm keeps only the previous centroids set which can be kept in the memory for very large data sets. That is why, as the size of the data gets bigger, k-means-ibd starts outperforming k-means-inbd in MapReduce parallel computing model. k-means-inbd on the other hand, has less complexity compared to k-means-ibd. Because k-means-inbd model keeps all data points' previous centroids, after several steps, calculation of the new centroids is taking $O(1)$ instead of $O(k)$ time most of the time. However k-means-inbd has an obvious space disadvantage. That is, when working with big data, in every iteration, all data points' centroids must be read and written to the disc, since they can not be kept in the memory. As it will be seen in Section 4, there is threshold value for data set depending on the overhead of writing big data to disc that dominates over k-means-inbd's improvement in the first part of the algorithm. That is why, we considered this algorithm to be the best for the serial implementation and for the parallel implementation with upper bound data size.

As demonstrated in [63], the worst-case running time of k-means is superpolynomial by improving the best known lower bound from $\Omega(n)$ iterations to $2^{\Omega(\sqrt{n})}$. That is, k-means always has an upper bound, therefore it always converges. So, because it always converges, the displacement speed of centroids must go to zero as iterations go to some finite number. Therefore, their speed must decrease, otherwise the algorithm cannot converge. Because centroids' speed decrease, the points that belong to particular centroid, tend to stay in that cluster.

3.1.5 Clustering Quality Improvement

Even though k-means is not the best performing data clustering algorithm, it is still by far the most widely-used one due to its simplicity, scalability and

convergence speed. However, depending on the initial starting centroid points, significantly different results can be obtained, so the algorithm is highly sensitive to its random centroid selection.

Meanwhile there is also another notable problem with k-means algorithm. When the number of cluster formations increase, the possibility of obtaining empty clusters (locations that do not have any data points associated with the clusters) also increases at each iteration. This becomes an unavoidable issue when $k >> 1$. This issue was not addressed thoroughly in most of the studies.

In this study we not only aim to improve the clustering quality of k-means, but also provide a methodology that can overcome "empty clustering" problem without jeopardizing the convergence speed, so the proposed model could be used for all clustering problems regardless of the number of data points, number of data dimensions and number of clusters. We achieved this by using an initial centroid selection algorithm based on a hybrid model that is a combination of evolutionary algorithms (Fireworks and Cuckoo search) and some additional heuristics.

Our model, $H(EC)^2S$, Hybrid Evolutionary Clustering with Empty Clustering Solution, consists of 4 parts: Representative Construction (RC) part, Enhanced Fireworks algorithm for Clustering (EFC) part, Cuckoo Search for Clustering (CSC) part and k-means part. Firstly, instance reduction is done to select representatives for existing data. This part does the main job to eliminate Empty Clustering problem, because in latter parts, the centroids are selected among representative points only. Secondly, in EFC part, the solution space is searched using Enhanced Fireworks algorithm. Thirdly, in CSC part, we construct new solutions and based on Cuckoo Search algorithm, place solutions to "host nests". Finally, we pass the selected best firework to the k-means and converge. The pseudo-code of the proposed model is given in Algorithm 5. Some notations are given in Table 3.1.

3.1.5.1 RC Part

The main purpose of RC part is to select "representative" instances among data which eliminate the instance count significantly. The important point is that, in RC part we can detect outliers and eliminate them also. Algorithm 4 shows the pseudo-code of RC-part. Lines 3-8 show the process of discretizing the values to find how many distinct data point there are at each dimension. Here $s_i[j]$ denotes the j^{th} dimension value of i^{th} data point. Equation 3.16 maps each data point to a representative value, where min_i and max_i represent min and max elements at i^{th} dimension respectively. To do this, it divides dimension range into F_i and finds in which portion the data is. Lines 9-13 find the representative values $rValue$ for each data point. Lines 13-16 show the elimination part of representatives. That is, if the number of belonging data points are less than the specified threshold, c_o, then the data points can be considered as outliers and representative can be deleted. Else, we find the average of all its belonging data points in each dimension and update the representative value. Finally, lines-17-20 show the case where the number of representative values do not fit into the memory, they are more than predefined threshold. In this case we either select another f function which is "more" decreasing or increase c_f or do both.

Algorithm 4 shows the main intuition behind RC part, which be can easily converted to MapReduce algorithm. That is, first we calculate distinct values in Mapper and send them to Reducers so that it can combine all distinct values in each dimension. Secondly, in Mapper we find each point's representative and send it to Reducer with the point itself. Reducer gets the representative and its points and calculates the average of the points. Figure 3.5 shows the overall intuition over the RC part, where romb-like rectangles are representatives and black points are data.

Algorithm 4 RC-Part

Require: $s_i \in S$ data points

Let H_k be the HashMap related with k^{th} dimension of data points, where $k = \{1, 2...c_d\}$

A_k be a vector, denoting the count of distinct values in k^{th} dimension, where $k = \{1, 2...c_d\}$

for all $s_i \in S$ **do**:

    for $j = 1$ to c_d **do**:

        $x_i[j] \leftarrow$ Round $s_i[j]$ to c_f decimal places.

        Put H_j a new entry with key=x_i, value=1. If already exists, increment value.

$A_i \leftarrow \sum_{e_j \in H_i} e_j.value, \forall (i = \{1, 2, ...c_d\})$

$F_i \leftarrow f(A_i) \forall (i = \{1, 2, ...c_d\})$

for all $s_i \in S$ **do**:

    Compute rID^k $k = 1, 2, ..c_d$ using Equation 3.16

    $R_{id} \leftarrow R_{id} \cup rID$

    $D_{(rID)} \leftarrow D_{(rID)} \cup s_i$

for all $rID_i \in R_{id}$ **do**:

    if $||D_{(rID_i)}|| \leq c_o$ **then**

        Data points belonging to representative rID_i are outliers, so eliminate rID_i

    else $rValue_i^k \leftarrow \frac{\sum_{s_j \in D_{(rID_i)}} s_j[k]}{||D_{(rID_i)}||}, \forall k = 1, 2, ...c_d$

if *representatives* are higher than predefined threshold (do not fit in memory) **then**

    $f() \leftarrow f(f())$

    Increase c_f

    Restart algorithm

Algorithm 5 $H(EC)^2S$

Run RC part

for $t = 1$ to $iter_{max}$ **do**:

    Run EFC

    Run CSC

Run k-means with the best firework

Table 3.1: Notations.

$rID_i^k \in R_{id}$	The ID of i^{th} representative data point, possibly int values denoting the range index, where k denotes dimension
$rValue_i^k \in R_{value}$	The actual value of i^{th} representative data point, where k denotes dimension
$f : R \to R$	Decreasing function from real numbers to real numbers
c_f	Constant denoting how many decimal places should data be rounded
max_d	Maximum data point in d^{th} dimension
min_d	Minimum data point in d^{th} dimension
N_i	Number of distinct data points in i^{th} dimension
F_i	Number of distinct data points in i^{th} dimension after applying f function to N_i
c_d	Dimension size of a data point
c_o	Threshold value for "representative" values to be used in outlier detection.
$D_{(rID)}$ or $D_{(rValue)}$	Set of belonging data points to representative rID or $rValue$
$E[i][j][n][k]$	Similarity matrix, that shows the shared data points count between i^{th} firework's j^{th} cluster and n^{th} firework's k^{th} cluster
$fw_i^j \in F$	Firework (a possible solution) of index i in set F, where k is the centroid of firework.
K	Number of clusters

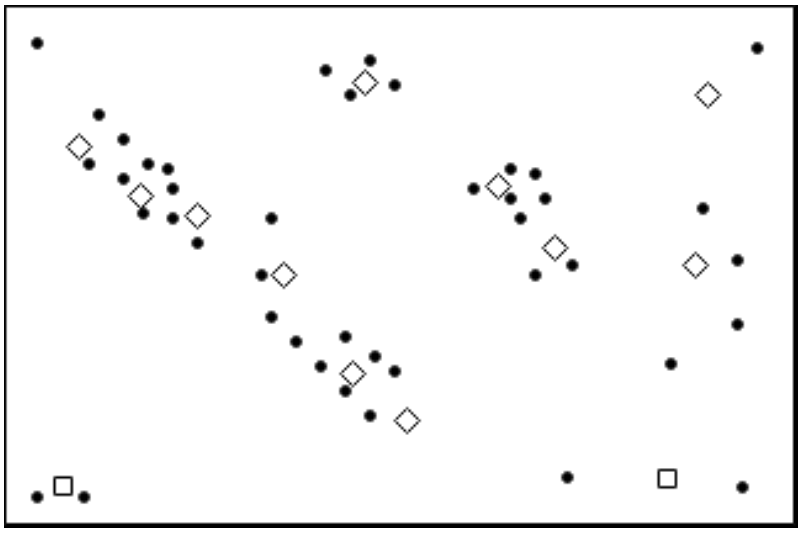


Figure 3.5: Demonstration of RC part

3.1.5.2 EFC Part

In this part, we use Enhanced Firework Algorithm (EFW) to search the solution space for "good" initial points. Initially, the model starts with random solution, converts it to *representative* using Equation 3.16 and calculates amplitude and and sparks locations. The $s_j[i]$ represents i^{th} dimension value for some $s_j \in S$, min_i and max_i are minimum and maximum values in i^{th} dimension. After that, while computing their fitness values, it also derives 4 dimensional similarity matrix E. The important point is that, we use the reduced data set with only representatives for selecting centroids, R_{value}, derived in RC part. Then, we find the amplitude and spark sizes in each dimension of firework and update fireworks. After updating, we convert them to representative values with Equation 3.16. The pseudo-code for this part is given in Algorithm 6.

The construction of firework fw as a solution is done via concatenating the k centroids. That is, because we seek a solution of k centroids to cluster data, we represent the solution, fw, as a combination of k cluster centroids. Equation 3.11 controls the A^k_{min}, the amplitude of firework. It starts with a higher value

initially to explore the solution space and diminishes gradually. Here t refers the number of function evaluation at the beginning of the current iteration, and $evals_{max}$ is the maximum number of evaluations. A_{init} and A_{final} are the initial and final minimum explosion amplitude, respectively. Equation 3.12 controls the amplitude so that it does not become too small. Equation 3.13 calculates the value of amplitude, where g is fitness function, $y_{max} = max(g(X_i))$ and $y_{min} = min(g(X_i))$ are the two constants to control the explosion amplitude and the number of explosion sparks, respectively, and ϵ is the machine epsilon. Equation 3.14 calculates the number of sparks.

Algorithm 6 EFC-part

1: Initialize random fireworks $fw_i \in F$
2: Calculate fitness of each firework $ft_i \in FT$ using Equation 3.15, where M is positive constant, s_i^j is j^{th} dimension value of s_i and c_j is j^{th} centroid which s_i^j belong to. While doing this, update the value of E array, such that, if data point belongs to fw_i^k and fw_j^m then, $E[i][k][j][m] \leftarrow E[i][k][j][m] + 1$
3: Calculate s_i, the number of explosions for each firework using Equation 3.14
4: Calculate A_i for every firework using Equation 3.13, 3.12 and 3.11

5: **for all** fw_i in F **do**:
6: **for** $j = 1$ to s_i **do**:
7: $\Delta X_i^k \leftarrow round(rand(0,1)) * A_i * rand(-1,1) \forall(k)$, calculate displacement, where $1 \leq k \leq K$ is cluster count
8: **if** X_i^k is out of bounds **then**
9: $X_i^k \leftarrow X_{min}^k + |X_i^k| \% (X_{max}^k - X_{min}^k)$
10: $X_i^k \leftarrow X_i^k + \Delta X_i^k \ \forall(1 \leq k \leq K)$
11: Perform k-means and update E array
12: Calculate fitness values of fireworks $fw_i \in F$ using Equation 3.15

$$A_{\min}^k(t) = A_{init} - \frac{A_{init} - A_{final}}{evals_{max}} \sqrt{(2 * evals_{max} - t)t} \tag{3.4}$$

$$A_i^k = \begin{cases} A_{\min}^k & \text{if } A_i^k < A_{\min}^k, \\ A_i^k & otherwise. \end{cases} \tag{3.5}$$

$$A_i = \hat{A} \cdot \frac{g(X_i) - y_{min} + \varepsilon}{\sum_{i=1}^{||F||} (g(X_i) - y_{min}) + \varepsilon} \tag{3.6}$$

$$s_i = M_e \cdot \frac{y_{max} - g(X_i) + \varepsilon}{\sum_{i=1}^{||F||} (g_{max} - f(X_i)) + \varepsilon} \tag{3.7}$$

$$J = \frac{\sum_{j=1}^{k} \sum_{i=1}^{n} \left\| s_i^j - c_j \right\|^2}{M} \tag{3.8}$$

$$rID^i \leftarrow \frac{s_j[i] - min_i}{\frac{max_i - min_i}{F_i}} \forall (i = 1, 2, ...c_d) \tag{3.9}$$

3.1.5.3 CSC Part

In this part, new fireworks are constructed and joined to firework set. We use E, 4 dimensional array to construct new solutions and use Cuckoo Search algorithm to select fireworks in F to pass to the next generation. To construct the new fireworks, k different centroids from different fireworks are found and joined. The important point is that, k different centroids cannot share any data points, because in a particle centroids do not have any common data points.

For example, Figures 3.6 and 3.7 show the procedure of creating new firework with k=3 and $||F||$=2. Figure 3.6 and Figure 3.7 are before-after situations of constructing new firework. In Figure 3.6, one firework is denoted with rectangle and another with triangle. CSC computes that centroids denoted with $0-0$, $1-1$ and $0-1$ do not share any common data points,where $f-k$ means k^{th} centroid of f^{th} firework. As a result, the new particle can be created as shown in Figure 3.7 with plus sign. Indeed, if this was a separate particle, it would have better fitness value than the existing ones.

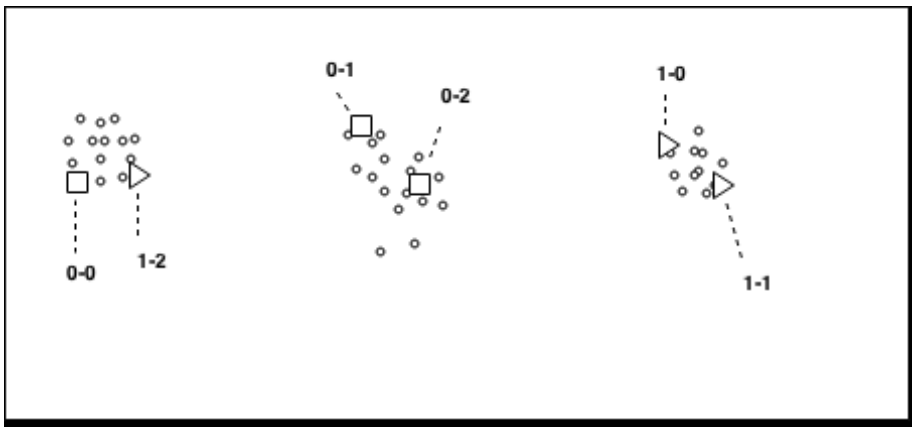


Figure 3.6: Before CSC part

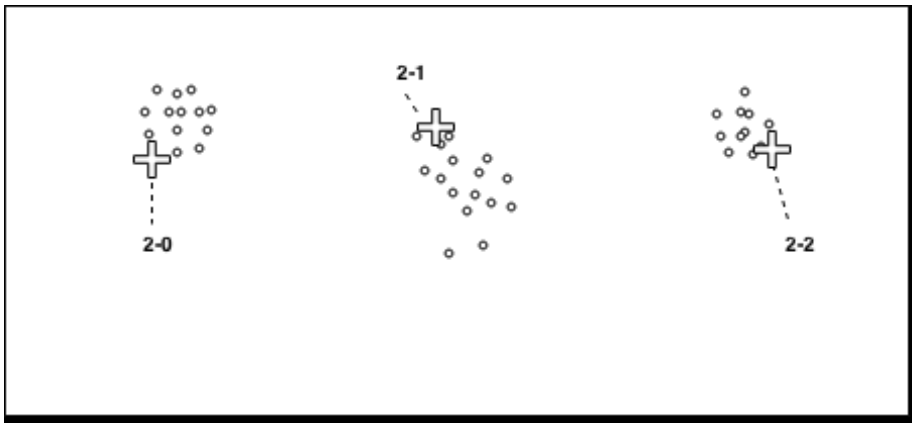


Figure 3.7: After CSC part

We use the similarity array E to form a graph consisting of vertices denoted with centroids and edges having value of that vertex's fitness value. Since the similarity array keeps the sharing data point count between centroids, if the shared element count is zero, then the corresponding centroids can be concatenated to form a new firework. As can be seen from Figure 3.8, the source node is the double lined circle. It has $||F|| * K$ child nodes, since that is precisely the total number of centroids within the firework set. The source node do not share any points with any of the centroids. As a result, some path, with the minimum sum of edge values, consisting of K consecutive nodes is found, where the incoming edge value of a particular node is its fitness value. Important point is that, the K consecutive nodes can be anywhere in the graph, might not necessarily begin

from the source node. To solve this problem, dynamic programming is used and all possibilities are tried like brute force, but in a "careful" way.

$$DP(k, c_p^k) = argmin(DP(k-1, n_p^k) + f_p^k, DP(k, n_p^k))\forall(n_p^k) \tag{3.10}$$

As can be seen from Equation (3.10), finding the minimum length k united nodes is implemented recursively, where n_p^k is the child node of c_p^k. The number of nodes needed and the current node are the two arguments given to function DP. Finding the minimum path length consisting of k nodes can be done by either taking the current node into consideration, running the same function for the rest of its child nodes with $(k-1)$ needed nodes and summing them, or not taking the current node into the consideration and running the function with K and all of current node's child nodes arguments, and giving the minimum of this results as an output. We used memoization in order to get the advantages of dynamic programming. That is, if some value of $DP(k, n_p^k) = c$ then it is stored, and after if it is needed, then the answer is retrieved in constant time and used.

After finding new solutions we use Cuckoo Search to place them. The pseudo-code for Cuckoo search is given in Algorithm 7.

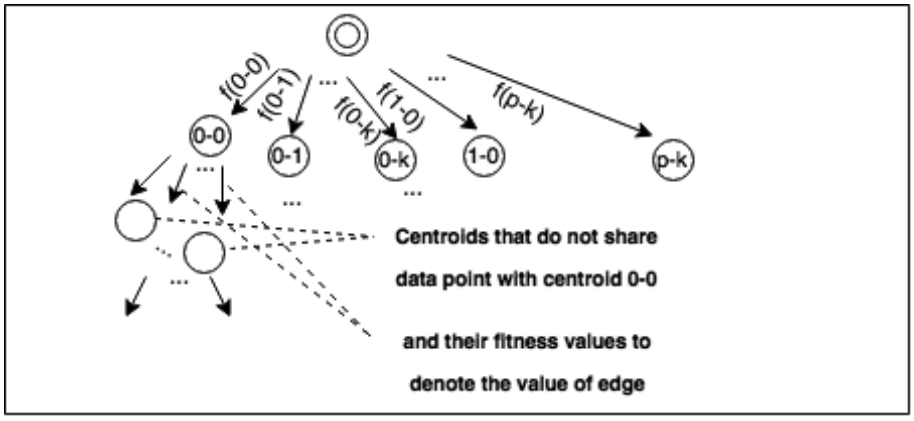


Figure 3.8: Intuition behind constructing new firework using graph.

Algorithm 7 CSC-part

Calculate objective functions $f_{obj}(fw_i) \forall fw_i \in F$ with Equation 3.15.
Get $n_i \in N$ new firework solutions with Equation 3.10
for $i = 0$ to $iter_{max}$ **do**
    Chose random n_i
    Chose random f_i
    if $f_{obj}(n_i)$ $isbetterthan$ $f_{obj}(fw_i)$ **then**
        Replace fw_i by n_i with some probability
Delete p_a fraction of F and build new fireworks randomly.

3.2 Optimization Part

Having clustered users, it is essential to construct a menu for each of the groups. The reason that we cluster users first is that to group them into similar clusters which can lead to better menus for each group. Obviously, it is not possible to make better menu for all users. Therefore, grouping them and making optimal menu for each group leads to better performance.

To explain overall system, we first present the problem definition with possible input and output constraints and optimization framework.

3.2.1 Before Optimization

Optimizing menu for specific needs is a challenge because it is vital to have understandable and usable menu after optimization part. There are some input and output specifications to be cleared before constructing a model. The steps that we have done before optimization, including in Clustering part:

- Mine user logs. The logs of user transactions have to be mined and for each user its corresponding customer profile must be derived.
- Cluster users. That was the process we have done in first part of solution.
- Derive menu tree. Here the menu screens can be thought as tree vertices

and relations between screens are edges in tree. For example, in Figure 3.9, the sample menu tree of some website is shown.

- Define optimization objective. For example, minimizing click count, maximizing menu visibility, minimizing overall session time and etc. can be objective for menu optimization. In this study we focused on minimizing click count problem.

- Calculate click counts for each menu screen for each cluster. The click count of each menu screen is not calculated as the number of clicks for particular menu item. Instead, our objective is to get actual aim of user. For example, if user clicks menu A then B and then C, and stays in A and B negligible time, then his/her aim is to reach menu C. So, counting menus A and B for this aim is worthless. Therefore it can be the case that the menu item under other menu item can have high click count than its "parent" menu.

- Put click counts on tree nodes for each cluster. In menu tree, we put click count of each menu screen to its relative vertex. That is, the value on the vertex corresponding to particular menu screen, is equal to that menu screen's click count.

Having input specifications our aim is :

- Construct a menu tree out of existing one such that the overall click count is minimized. For example, if menu C is under menu B and menu B is under menu A (A→B→C), and if menu C is very usable (having high click count), then to reach this menu users click more. If we reposition menu C is nearer position to main menu (A), then overall click count can be minimized.

- Ensure the usability and understandability. To reconstruct the menu in optimum way, one can say that using Huffman coding [64] and placing the most clicked menu on top and reconstructing all subtree in this manner gives the best solution. This is true but how about usability? For example, we have in original menu structure "Fox","Cat" and "Elephant" is under

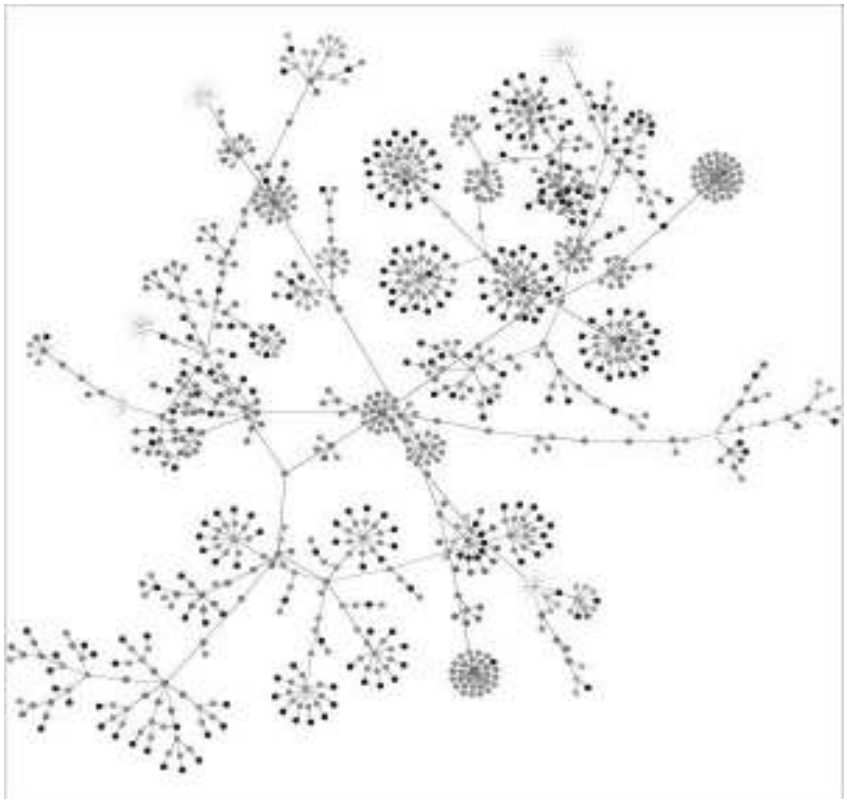


Figure 3.9: Sample component design of web page

the "Animal" menu. If "Cat" is clicked much more than others, we cannot be put "Cat" on top of "Animal" menu, because this would mislead the users and affect usability negatively.

- Ensure the additional constraints. Some devices can have constraints such as maximum menu buttons for each page. For example, in phone or ATM menus the space constraints are important issue. Our model handles it considering each menu screen's maximum children menu pages.

Figure 3.10 gives sample menu structure of ATM, where rectangle boxes represent menu item, the numbers above boxes represent their ID, the ones below boxes represent click counts for particular menu item and black boxes represent "leaf" menu items, meaning last menu item in the click path. Moreover, we assume that the menu shown in Figure 3.10 can have at most 3 children per node.

The optimal version of that menu based on the click count is given in Figure 3.11. It is clear that, the menu is optimum when it conserves both usability and understandability. Meanwhile, in this particular case, finding the optimal menu for Figure 3.10 is not very hard. However, considering very large menus for complicated systems or enormous networks, the optimization is not an easy task.

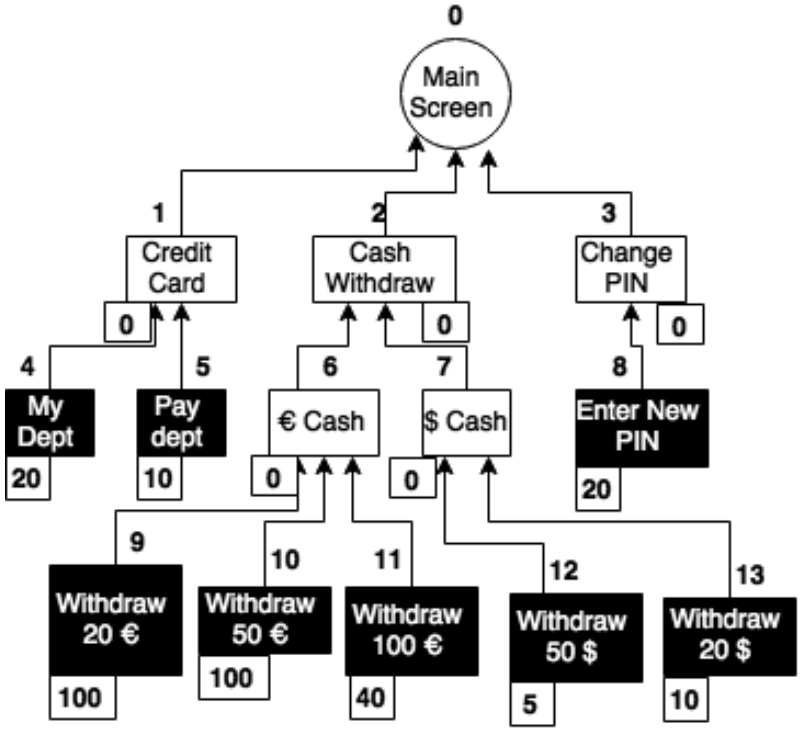


Figure 3.10: Sample menu structure of ATM before optimization

3.2.2 Optimization Framework

To optimize user menus we assume that, the user logs are mined in an efficient way, they are clustered and click counts of menu items for each cluster is known.

Using MIP framework we guarantee the optimum. The key concepts are represented in Table 4.1 and in Figure 3.12. In this solution, we represent the menu as tree and show the existing problem as a network flow problem. The flow generated in each node is equal to overall click count on particular node and aim

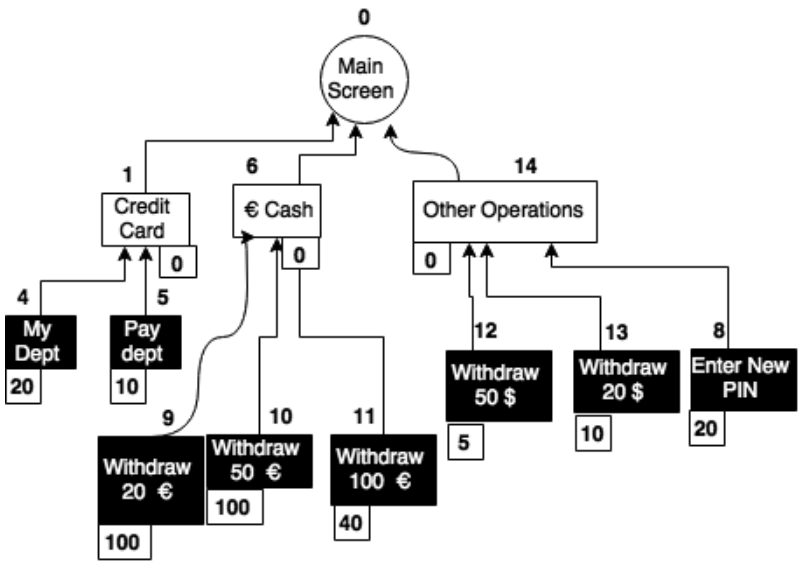


Figure 3.11: Sample menu structure of ATM after optimization.

is to minimize the overall flow. Figure 3.12 shows the original menu and dashed lines represent possible connections between nodes that can be selected while optimization. As can be seen from Figure 3.12, we introduce 2 new node types: optimizers (triangle-like boxes) and combiners (romb-like boxes). The details and their usage is given below:

- Optimizers - These are menu nodes introduced in this study to bring best leaf node or menu subtree to upper levels. That is, the optimizer has possible connections to all "grandchildren" nodes of its parent, but selects only out of them. For example, in Figure 3.12, nodes with ID=14,17 and 18 are optimizers and all lower level nodes with same parent (Main Screen) are connected to them. The optimizer selects the best option as there can be only one child for each of optimizers. Optimizers can have in all levels of a menu tree but the leaf level. For simplicity, in Figure 3.12, we put them only in the first level. The important point is that, optimizers can connect to not all low level menu nodes but only same-parent menu nodes with itself. Moreover, the cost of connecting to optimizer is zero. This constraint is put to encourage nodes to go to upper levels with minimum cost and the

Table 3.2: Terminology for MIP Formulations

Variable	**Description**
S (Given)	Set of all original menu nodes, excluding optimizers and combiners.
S_{OPT}(Given)	Set of Optimizer menu items (triangles in Figure 3.12)
S_0(Given)	The root or top level menu item. (Node with ID=0 in Figure 3.12)
S_{COMB}(Given)	Set of Combiner menu items, the ones shown with romb in Figure 3.12)
C_i(Given)	Integer that shows click count for i^{th} node.
IN_i (Given)	Set of candidate incoming nodes to i^{th} node.
OUT_i (Given)	Set of candidate outgoing nodes from i^{th} node
L^i_{IN}(Given)	Constant number that shows the number of child nodes of i^{th} menu node. This number is maximum number of incoming nodes to i^{th} node.
L^i_{OUT} (Given)	Constant number that shows the number of parent nodes of i^{th} menu node. This number is the maximum allowed number of outgoing nodes from a particular node i.
f_{ij}(Variable)	Amount of flow (click in this case) between menu nodes i and j
a_{ij}(Variable)	Boolean variable which denotes if there exist a flow between the nodes i and j
d_{ij}(Given)	Integer value denoting the weights of arcs. In general, all weights are equal to one, but the ones incoming to S_{OPT} are equal to 0
M(Given)	Large constant with respect to given parameters in problem.

optimizer will select the best option to minimize overall system's cost.

- Combiners - These are menu nodes introduced in this study to combine same level nodes under one menu node and take them one level below. For example, node with ID 16 is combiner. All same level nodes are possible child nodes of combiner. The main point is that, if some nodes in the same level are chosen to go downwards, they are collected under the combiner node with name like "Others" and their previous places are taken by another menu nodes from downwards.

It is assumed that whatever the size of menu is, we have quite powerful machine to compute its optimum form. The model consists of several linear equations given below:

$$a_{ij} \in \{0,1\}\ \forall i \in S, \forall j \in OUT_i \tag{3.11}$$

$$f_{ij} \geq 0\ \forall i \in S, \forall j \in OUT_i \tag{3.12}$$

$$\sum_{j \in OUT_i} f_{ij} - \sum_{j \in IN_i} f_{ji} = \begin{cases} -\sum_{i \in S - S_0} C_i, & i = S_0 \\ C_i, & \forall i \in S - S_0 \end{cases} \tag{3.13}$$

$$f_{ij} \leq M a_{ij}, \forall i \in S, j \in OUT_i \tag{3.14}$$

$$\sum_{j \in OUT_i} a_{ij} \begin{cases} \leq 1, & \forall i \in S_{OPT} \cup S_{COMB} \\ = L^i_{OUT}, & \text{otherwise} \end{cases} \tag{3.15}$$

$$d_{ij} = \begin{cases} 0, & \text{if } j \in S_{OPT} \\ 1, & \text{otherwise} \end{cases} \forall i \in S, \forall j \in OUT_i \tag{3.16}$$

$$\sum_{j \in IN_i} a_{ji} \begin{cases} \leq 1, & \forall i \in S_{OPT} \\ \leq L^i_{IN}, & \text{otherwise} \end{cases} \tag{3.17}$$

$$\text{Minimize} : \sum_{j \in OUT_i} d_{ij} f_{ij} \forall i \in S \tag{3.18}$$

Equation 3.11 shows that a_{ij} is a boolean variable for all possible arcs defined in the system. Constraint 3.12 represents the non-negativity of flows in our model.

Here, flow is the click count of a particular node. Equation 3.13 provides the flow balance in the system. There can be 2 separate cases. If the node is a sink node, that is, the one at the top level, then all incoming flow is equal to the sum of all flows generated in other nodes, since flows are made from clicks. Otherwise, the flow generated in particular node is the difference between incoming and outgoing flows. In equation 3.14, we are converting flows to binary variables to be used in later equations. Equation 3.15 shows the constraint related with outer arcs from a particular node. That is, there must be exactly L^i_{OUT} number of arcs going out of i^{th} node, if the menu node is neither S_{OPT}, nor S_{COMB}. This means that, a particular node in the menu must be connected to the menu and it cannot be lost even it has no clicks. But in the case of S_{OPT} and S_{COMB} nodes, they are free to be connected to the tree or not. We are not forcing them to do so. Equation 3.16 shows the constant d_{ij}, which represents the weights of arcs. We say that every arc in a tree has an equal weight of 1, except the ones incoming to S_{OPT} menu nodes. Since we want to encourage other nodes to connect to these nodes, their weights are 0. Equation 3.17 does the same for constraining the number of incoming arcs to a particular menu node. This can be thought as the maximum number of menu items under a particular menu item. We constrained S_{OPT}'s incoming nodes to maximum 1 node, as we are allocating S_{OPT} nodes itself and taking more than one incoming node would not optimize the system efficiently. Equation 3.18 shows our objective function. This is simply the minimization of flows multiplied by the arc weights. This can also be considered as the cost of a flow.

The important point is that, our model does not enforce using combiners and optimizers. If it is optimal, the system uses them in some or all levels of the menu tree. Another important point to mention is that, the click count of non-leaf nodes is not computed as regular. For example, there can be the case that, for one menu node m_1 click count is c_1, and for another menu node m_2 click count is c_2, where $c_2 > c_1$ and m_2 is child node of m_1. Here, for non-leaf menu nodes, we do not compute click count as regular. If the user stays at a particular menu item for longer than some threshold time, then we add click count for that particular

menu node, else we consider that user's purpose was not actually looking for this page; therefore, we do not add click count.

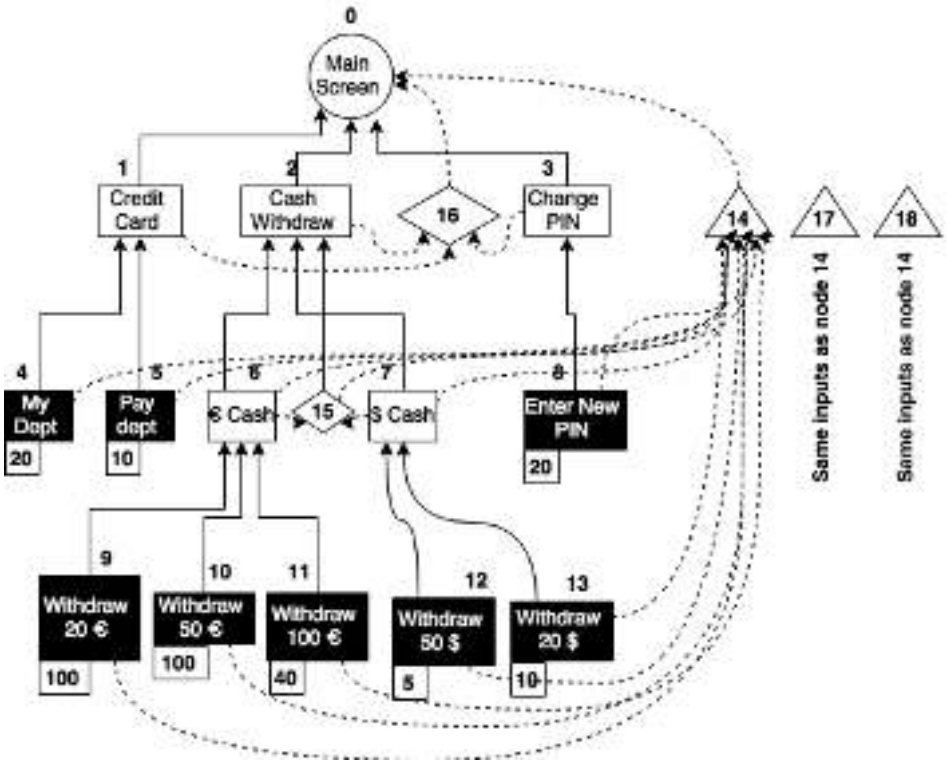


Figure 3.12: Sample menu structure of ATM after optimization.

4. EXPERIMENTS AND ANALYSIS

Because our solution is multi-objective, we grouped the experiments from general to specific to menu optimization. Firstly, the clustering performance improvement experiments are presented. Then clustering quality improvement experiments are presented. Finally, considering clustering results are ready, we optimize menu structure and show results.

4.1 Experiments on Clustering Performance Improvements

The experiments were conducted both in serial and parallel environment. MapReduce framework of Cloudera's Apache Hadoop distribution was used for parallel environment. The environment consisted of 17 connected computers with $100Mbit/s$ Ethernet. Each computer had Intel i7 CPU and 4GB RAM capacity. Among the 17 computers, 16 of them were worker nodes and 1 was the master node.

Two different data sets were used to run the experiments. First data set (DS-1) was, "Individual household electric power consumption Data Set" [1] and the second one (DS-2) was, "US Census Data (1990) Data Set" [2]. The lengths of feature vectors of DS-1 and DS-2 are 7 and 68 and the size of data sets are 2075259 and 2458285 instances respectively. Both data sets were divided into different number of clusters and the algorithms run with different initial centroids. Finally, we chose α threshold to be 0.15 in experiments.

We have performed numerous experiments both in serial and parallel environments. We compared our proposed improvements with the models proposed in

[1] https://archive.ics.uci.edu/ml/machine-learning-databases/00235/
[2] https://archive.ics.uci.edu/ml/machine-learning-databases/census1990-mld/

[62], [65] and standard k-means - (k-means-s) [61]. The complexity and efficiency of the models described in [62] and [65] are mainly the same. Therefore we implemented the model shown in [62] to compare with our proposed algorithms. As the authors of [62] called their model as enhanced k-means, for simplicity we called their model as k-means-e.

Before discussing the results, one important thing is that, there is no k-means-s in figures, because the graphs show relative results with respect to k-means-s. Since in all of the fore-mentioned models we are trying to achieve improvements over k-means-s, all graphs shown in this section used k-means-s performance as the basis. This is accomplished by dividing the result of the particular model running time by the running time of k-means-s. This can also be considered as a normalization.

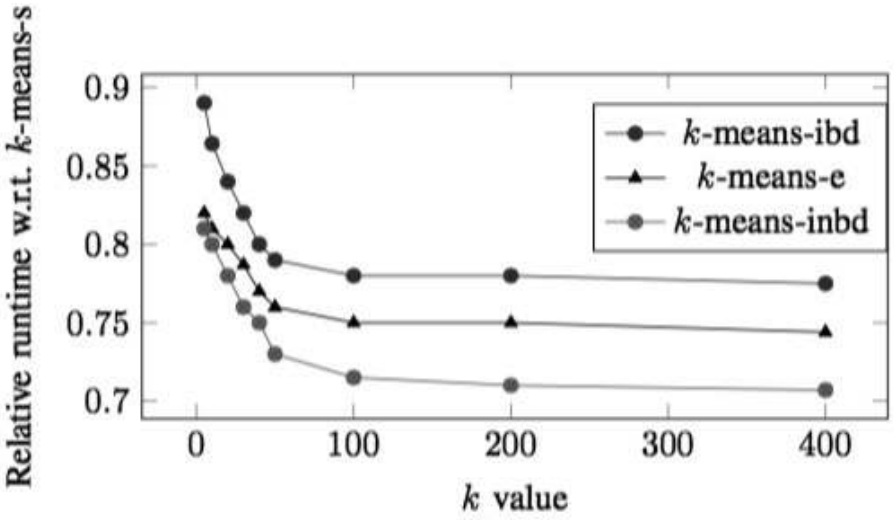


Figure 4.1: Comparison of three models in serial environment with DS-1.

Figure 4.1 shows the comparison of our proposed models and k-means-e [62] in terms of their efficiency towards k-means-s [61] with DS-1 in the serial environment. It is clear that k-means-ibd is less efficient compared to the other two models. As stated above, all proposed models consist of two parts: first part and second part. Here k-means-ibd mainly takes advantage of the improvement

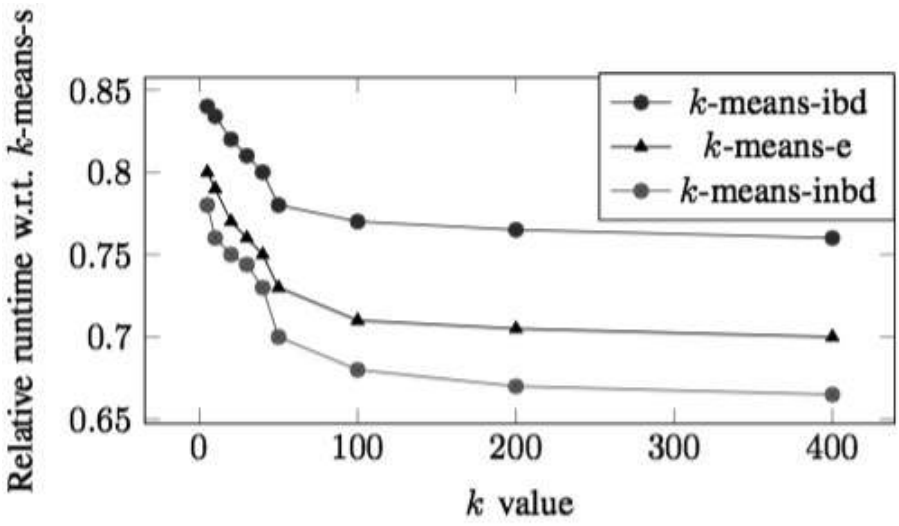


Figure 4.2: Comparison of three models in serial environment with DS-2.

of the second part when compared to k-means-s. k-means-e also takes advantage of the improvement of the first part, when finding the nearest centroids. However, k-means-inbd is improved both in the first and the second part, that is why, it performs better than the other models. In general, it is obvious that when the data is small compared to the memory size and in the serial environment, first part of the models dominates the second part. That is why, even though k-means-ibd performs better than k-means-s, it is still slower than the other two models.

Figure 4.2 shows the same procedure as Figure 4.1 with DS-2. It is clearly seen that the overall concept is pretty much the same. Meanwhile, all models more or less have improved their performance slightly by decreasing their computation time. This is due to the fact that the feature vector size in DS-2 was 68, whereas it was 7 in DS-1. However, k-means-inbd and k-means-e improved their computation time more than k-means-ibd, when compared to Figure 4.1 because k-means-e and k-means-inbd benefit computationally over k-means-s in part-1 which is directly related to the vector size, more than k-means-ibd. As the vector

dimension increases, k-means-inbd and k-means-e have more dominance over k-means-s, since the first part of the proposed models have dominance over the second part in the serial environment.

The performance improvement over standard k-means increases with larger k values due to the increase in the number of iterations to converge, as seen in Figures 4.1 and 4.2. As the number of iterations increase, we have much more benefit using our proposed models compared to standard k-means.

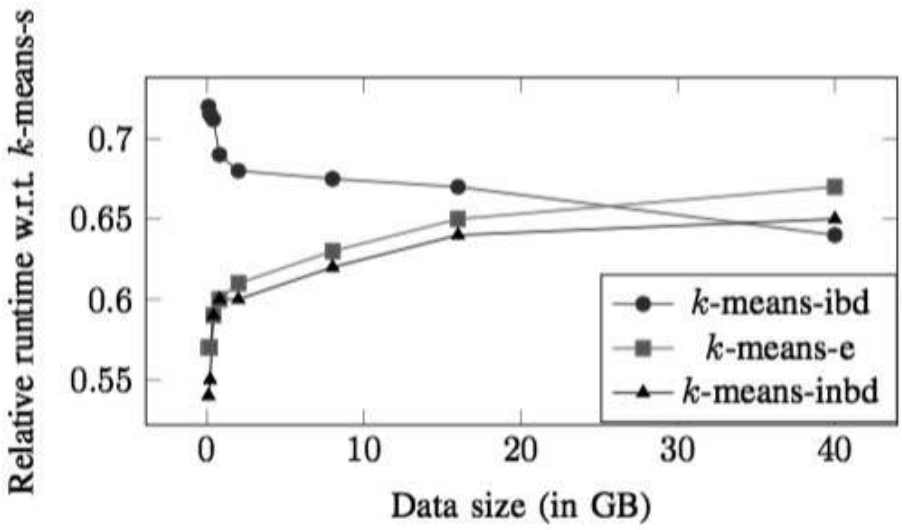


Figure 4.3: Comparison of three models in parallel w.r.t. data size.

Figure 4.3 shows the comparison of the proposed models over k-means-s in the parallel environment. We used Cloudera's Hadoop distribution with 17 nodes in this experiment. The main purpose of this experiment was to find the threshold value for the size of the data set where k-means-ibd starts outperforming k-means-inbd. Therefore we simulated DS-1 to have a larger data set. As stated during the analysis of the algorithms, the main disadvantage of k-means-e and k-means-inbd is their necessity to keep all data points' previous centroids. If the serial environment is used with a data size that is less than the memory size, they can be kept in the memory. However with the increasing data size, it will not be possible to achieve that. In MapReduce implementation, k-means-e

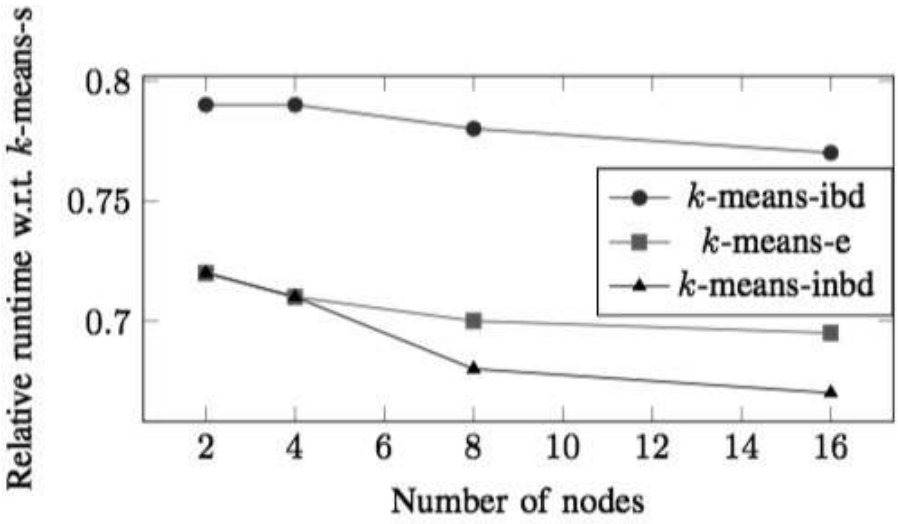


Figure 4.4: Comparison of three models in w.r.t. node number.

and k-means-inbd change the data set with their modified centroids and in every iteration output the result. That is, in every iteration, stated models read $O(2n)$ data points and output $O(2n)$ data file size. As the size of the data file gets larger, the dominance of the improvements keep decreasing due to the increasing I/O time to output and write to the large disc files. Moreover, reading time is also increased due to extra $O(n)$ data read in each iteration. As the iteration number gets larger to converge, this drawback becomes a major issue for k-means-s and k-means-inbd. We can see that in Figure 4.3 for DS-1, k-means-ibd keeps getting slightly better as the data size increases, because of the improvement in I/O and in the *reducer* side of MapReduce. However, k-means-e and k-means-inbd have a deteriorating performance with the increase in the data size due to the reason stated above. Another notable observation is, k-means-e's data size threshold being less than k-means-inbd. This is mainly due to the fact that it has no improvements in the reducer side and all points are sent from the mapper to the reducer in every iteration. However, in k-means-inbd as well as in k-means-ibd only those that have changed their cluster are considered for processing in the reducer. The reducer and the partition phase take longer when the data set becomes larger. However, as k-means-ibd does not send all data points from the

mapper to the *reducer*, it takes advantage of the reducer phase improvement and this advantage becomes more significant with increasing data size. Also, k-means-ibd has an I/O advantage over the other models, because of getting rid of reading $O(n)$ and writing extra $O(2n)$ data in each iteration.

Figure 4.4 shows the comparison of the proposed models' computation time over k-means-s with different number of nodes using MapReduce. In Figure 4.4, we used DS-2 with its size simulated up to 400MB and $k = 100$ in this graph. We simulated the data set in order to see real outputs that were less influenced by the network overhead. Here it is seen that k-means-inbd and k-means-ibd have increasing performance improvement over k-means-s as the number of nodes in the cluster increase. The main reason for this is, if we have $O(x)$ improvement in one node and if we distribute the job to m nodes, we will have $O(x * m)$ improvement, not considering the network overheads. However, the improvement of k-means-e is less than k-means-inbd as the number of nodes increase. Again the reason is k-means-e's not getting the advantage of the reducer side improvement. Overall picture is the same for data set 1, but again, have slight improvements in k-means-ibd and more improvements in k-means-e and k-means-inbd. The reason can be explained as follows: k-means-inbd and k-means-e have mapper improvements, that is why, they outperform k-means-ibd in small data sizes (in this particular case, with a data size of 400MB). As we increase the size of data set, the relative performance of k-means-ibd increases and after the size threshold value, it outperforms other algorithms.

Figure 4.5 shows the percentage of data points that was processed in the *reduce* step in both k-means-ibd and k-means-inbd for $k = 5$. This case was chosen as an example to demonstrate the general concept in a realistic environment. In that particular graph, it can be observed that after a certain iteration, all data sets had decreasing number of operations performed at each following iteration. This was our main motivation to the improvement achieved on the second part of the proposed models, namely, processing of only those points which altered their clusters to compute the new centroids. Since not all of the points are

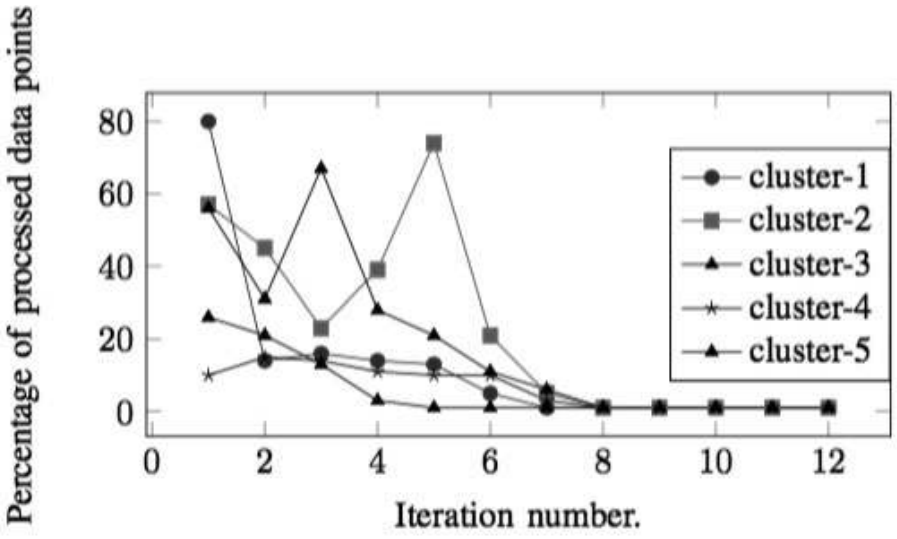


Figure 4.5: Percentage of points changed in clusters in reduce step for k=5.

considered in finding the new centroid (which is not the case in standard k-means), after several iterations the number of points that has changed their cluster decreases drastically. If this graph had included k-means-s, all lines would be ($percentage = 100$) straight lines; that is, no matter which iteration was carried, all *reducers* process all points. However, in our proposed models, as the iteration numbers increase, clusters tend to converge and the number of operations reduce geometrically. For simplicity, we show the graph until the 12^{th} iteration, because after that, points in all the clusters change less than 1 percent until convergence. It is obvious from the graph that, as the number of iterations increase, our proposed models demonstrate better efficiency.

Figure 4.6 shows the number of iterations needed to satisfy threshold α and the ones needed to converge. This is the point where k-means-ibd switches from Algorithm (1) to Algorithm (3). Since k-means algorithm is greedy, it converges at the first local minimum. Therefore after a few steps, the clusters tend to be stable as their centroids tend to move more slowly. So, with increasing k values, we have more iterations after threshold. This means that we can take more advantage on *reducer* side at each step.

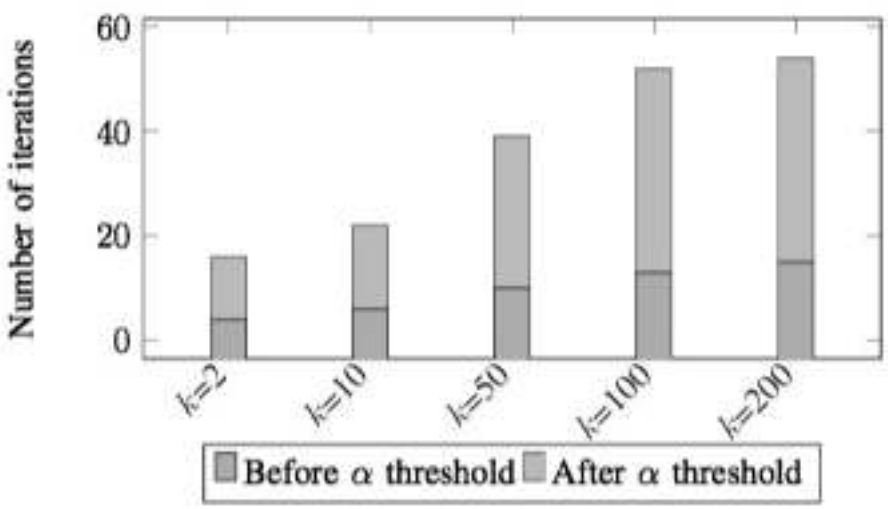


Figure 4.6: Number of iterations before and after $\alpha = 0.15$ threshold with DS-2.

4.2 Experiments on Clustering Quality Improvements

We used two data sets for test our model. First data set, DS1 [66], is public data set. Second data set, DS2, is the ATM logs data set, we used for case study. First data set contains 17389 instances and 16 attributes. DS2 contains 20 million instances consisting of 50 attributes. DS2 contains the user profile vectors for ATM customers. To construct a vector, we used the count of the customers' clicks for each transaction for a period of time. To test our results, we used Cloudera's Hadoop distribution on 5 Intel i5 machines.

We used [67], [68], [69] and our model to make comparisons. For simplicity, we called them rk-means, ck-means, $FGKA$ and $H(EC)^2S$ respectively. Because the fitness function, Equation 3.15, uses minimization objective, the better clustering quality is achieved with lesser fitness value.

Figure 4.7 shows the effect of increasing k value to the models with DS2. Here, the models with lesser fitness values are better because fitness function's, Equation

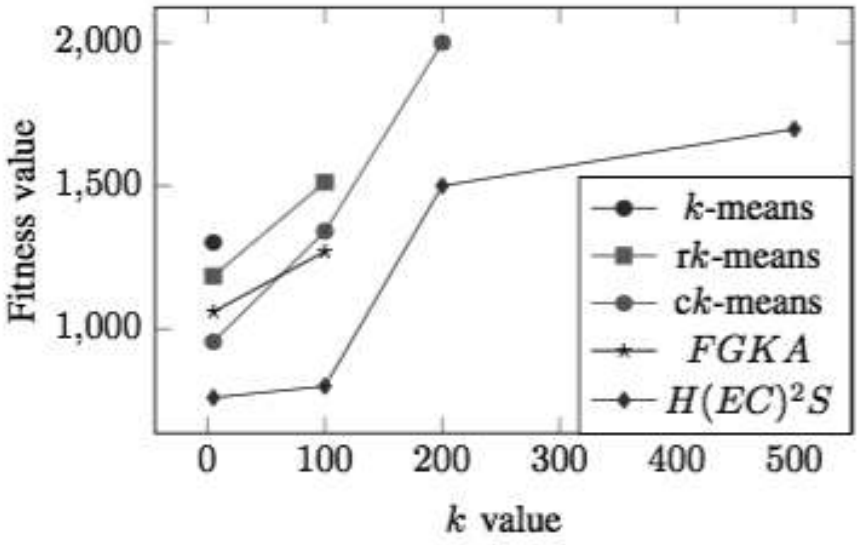


Figure 4.7: Effect of k to fitness function.

3.15, objective is minimization. As can be seen, the standard k-means cannot survive to k=100 with DS2, because of empty clustering problem. The same can be said to $FGKA$ algorithm after k=100 and etc. The reason is that, DS2 data set is very crowded in specific places and solution space is very large. That is, most of the users do same operations and dividing them into $k >> 1$ clusters is very difficult. The important thing in $H(EC)^2S$ model is that as k value increases, the gap between other models increases linearly. This is because of CSC part, in which new solutions are found and placed into fireworks. So, as k gets bigger, in CSC part, we have many solutions to select among, one of which can be the optimum solution as well. Table 4.1 shows the numerical results based on this experiment. Here $-$ sign indicates that the model could not find the solution because of empty clustering problem.

Figure 4.8 shows the dependence between data size and execution time implemented on DS1 on Hadoop environment based on 5 nodes. We simulated the data to increase its size. Because there are not scalable versions of other models given in Figure 4.7, we only considered k-means parallel [61] and $H(EC)^2S$. It is clear that k-means performs better in terms of performance time because of its

Table 4.1: Fitness values of different models w.r.t. k value.

	k-means	rk-means	ck-means	FGKA	$H(EC)^2S$
k=5	1302	1185	955	1060	760
k=100	–	1512	1341	1270	800
k=200	–	–	2007	–	1500
k=500	–	–	–	–	1698

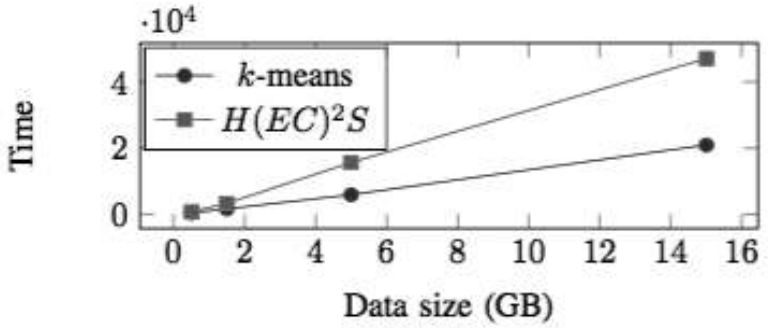


Figure 4.8: Effect of increasing data size to execution time.

greedy nature. However, $H(EC)^2S$ does not increase exponentially with respect to k-means. This is because of its scalability.

Figure 4.9 shows the dependence between c_o and fitness function in $H(EC)^2S$ model on DS2. Because c_o is outlier threshold, based on data set, it can vary. DS2 is very crowded data set in specific places of solution space. So, making outlier threshold bigger, can eliminate the necessary fireworks too. Therefore, after 1000 the fitness value gets worse.

Figure 4.10 shows the dependence between fireworks set size and fitness function value on DS1. As the fireworks set gets bigger, the probability of finding the new better solutions also increases. However, after some certain value, depending on the data set, the increase may not continue. The same applies to this Figure. We can conclude that the fireworks set size for this data set can be chosen nearly 50, because increasing further increases the time complexity also.

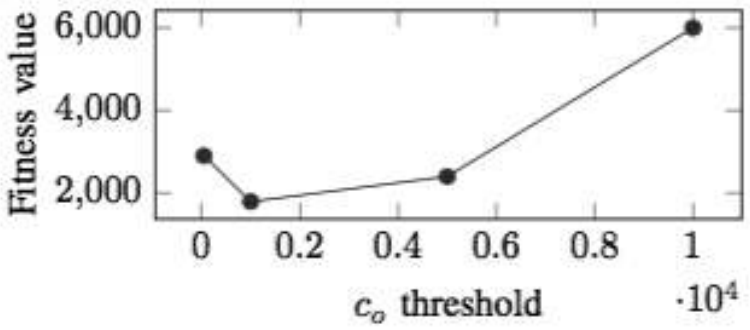


Figure 4.9: Effect of threshold level c_o on fitness value.

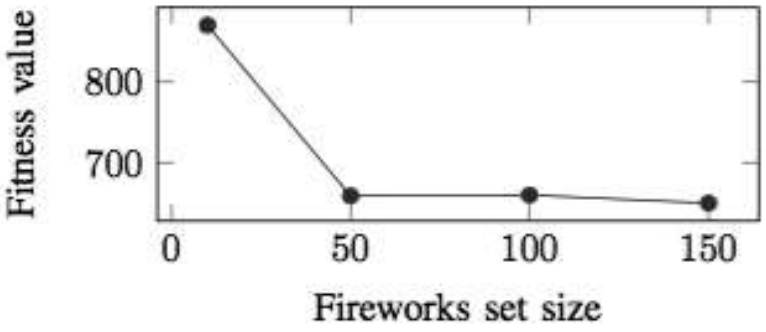


Figure 4.10: Effect of fireworks set size on fitness value.

4.3 Experiments on Numerical Optimization

We obtained initial results for only one ATM out of 2000 machines and we divided users into 3 profiles using the clustering algorithm we proposed before. Our data size is $4 * 10^7$ ATM logs.

We used MapReduce framework [1] to mine and process the data and CPLEX [70] to find the optimum solution using MIP [71]. Overall structure of the system design is as follows:

1. First we mine large amount of data using MapReduce and cluster the users.
2. Second, for each cluster we get the click counts of the leaf menu items using MapReduce.
3. Third, we find the optimum menu structure for each user group using the method explained in Section 3.2.2.

As the first and second steps are not the focus of this study, we try to concentrate on the third step. However, in the first step we choose k, the number of clusters to be 3 and separated users into 3 categories according to their actions in ATM, which are:

1. C1, Users that mostly took cash from ATM and look to their balance.
2. C2, Users that mostly took cash from ATM and almost never look to their balance.
3. C3, Users that do advanced operations in ATM such as money transfer, credit card operations, automatic bill payment and etc.

Our menu structure that we wanted to optimize consists of 35 menu items. The time taken to optimize this count of nodes is low (*i.e.*, it was less than a second).

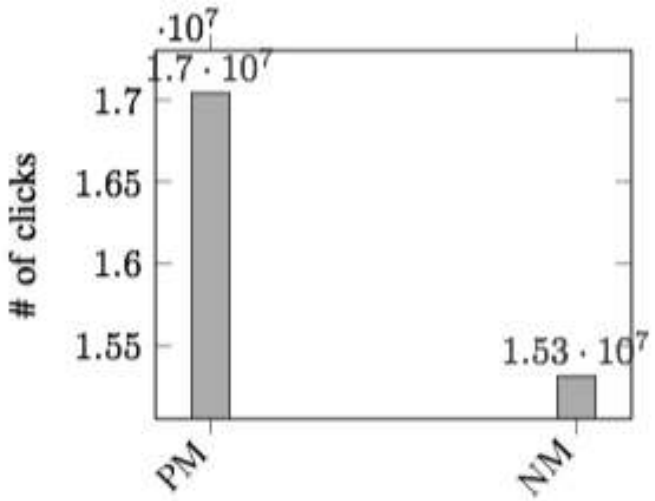


Figure 4.11: Overall number of clicks with new model (NM) and previous model (PM).

However we are working on heuristic methods to optimize huge number of menu structures. We tested our results with two methods. In first method (FM), we mined all logs of ATM derived a conclusion about overall click count with new menu structure. In second method (SM), we chose random person from the customers of the bank, and asked to use our app developed in iOS. SM is subset of FM because FM includes all user transactions in the past 18 month for a particular bank.

Figure 4.11 shows overall count of clicks with new (NM) and previous (PM) models. There is approximately 10% click reduction in the new model.

Figure 4.12 shows the user profile click gains. Here NM-k shows the k^{th} profile users with new model and the PM-k shows the same with previous model. Because there was no user profiling in previous model, we did the same procedure (clustering of users) to the previous one to compare the results. The results show that, there is a significant reduction of clicks in C1 and C2. This is because of the fact that, C1 and C2 clusters are more specific profiles. C3 cluster users do all kind of operations and mostly advanced ones. So, as the clusters get more specific, the menu gets more efficient optimized structure. That is, getting single

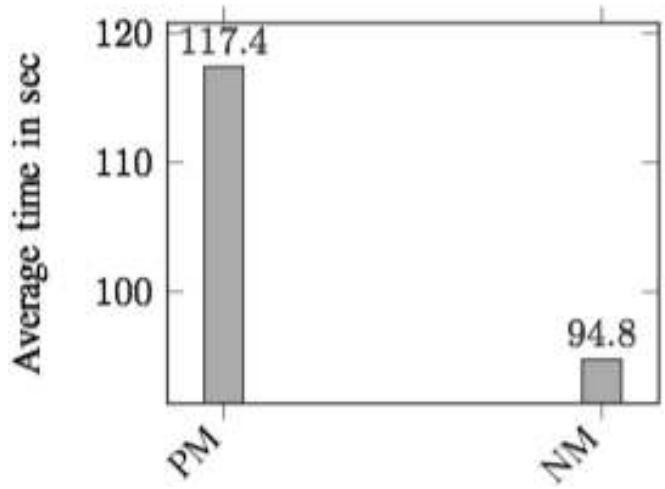


Figure 4.12: Profile based comparison of number of clicks.

optimum menu for all users is harder and inefficient than getting the same for clustered or profiled users.

Figure 4.13 shows the average session time with previous and new models which showcases the gains brought by the proposed MIP model. As the users are more efficiently clustered, the reduction in completion time will higher, which is among our ongoing research efforts. Figure 4.14 shows the percentage of users that used our new optimizer menu items which was created by our model. As can be seen C1 and C2 users tend to use more, because they represent more specific profiles.

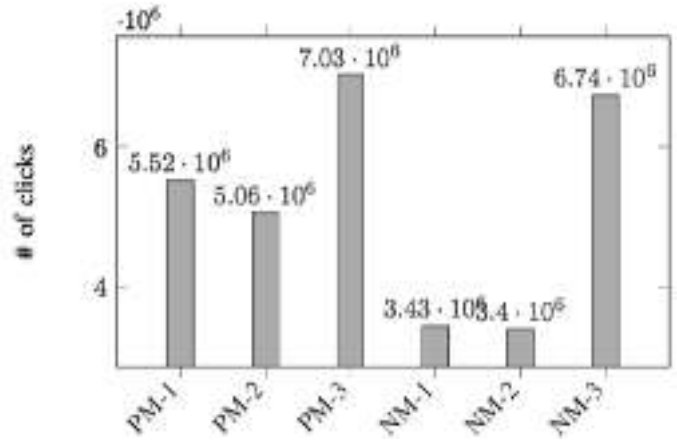


Figure 4.13: Average session time (in sec) on ATM with new model(NM) and previous model (PM).

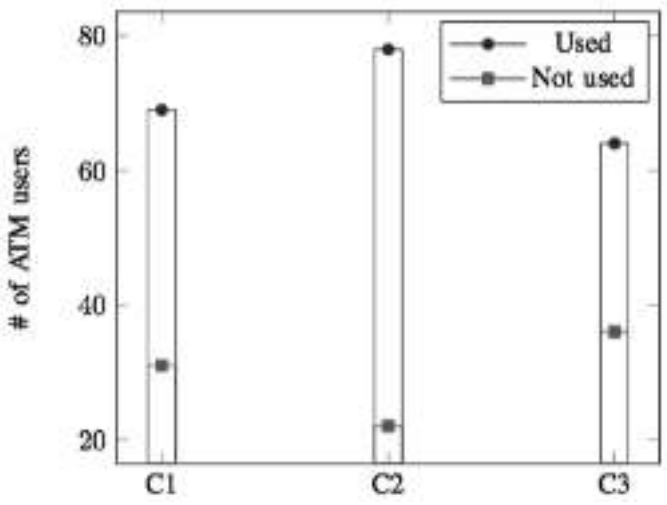
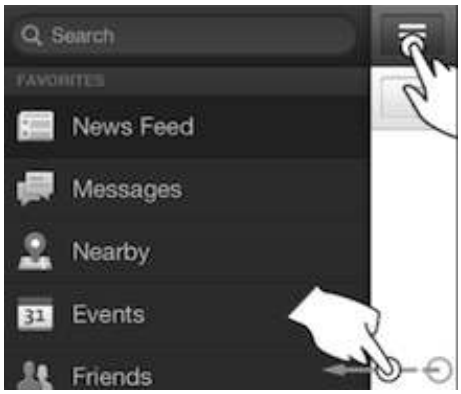


Figure 4.14: Percentage of users used our model's optimization menu items (S_{OPT}).

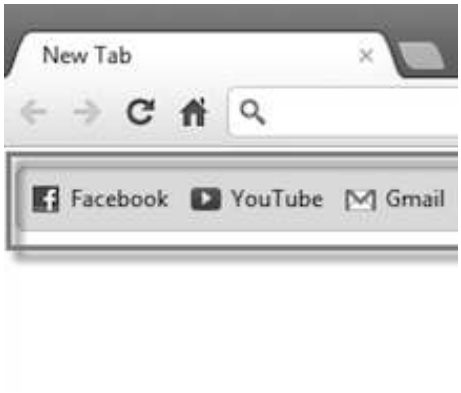
5. APPLICATION AREAS

Main applications of our work is related with UI included systems. For example, as shown in Figure 5.1a, when customers use X operating system, after some time based on their actions the system can change its structure, because they can be seen as a tree of menu items. Moreover, there are many web sites that do research to satisfy their customers, try to learn their behaviors and satisfy them. Considering that web pages also have menu tree, customizations can be made based on group of users of for single user separately. Another possible application that we made was the application to ATM menus. In this research, as indicated in [72], we reached significant improvements. Yet another application is for mobile device menus, as shown in Figures 5.1b and 5.1d, because they are getting more and more popular. Yet another application is vehicle routing problem. In some cases, vehicles cannot go to any places and must go through several nodes before going to the next. This can also be converted to menu tree problem and solved with our methods proposed. Moreover, optimum bookmark browsers can be developed using this model as shown in Figure 5.1c. That is, based on user actions in the web, bookmarks can be selected automatically.

(a) Usage of solution in operating systems, MAC OSX in this case.



(b) Usage of solution in mobile slide-menu.



(c) Usage of solution in web-browsers, chrome in this case.

(d) Usage of solution in mobile tab bars.

Figure 5.1: Some possible usages of proposed model

6. CONCLUSION

In this study, we tried to solve menu optimization problem with a deterministic solution and at the same time developed new models for clustering. So, we reached 3 main improvements to consider them all as a whole solution.

Firstly, we suggest a solution of improvement over the standard k-means clustering algorithm. In particular, instead of using the full data set in the centroid updating step of the algorithm, only the data points which will change their cluster (by associating themselves to a different cluster centroid) would be considered. This adjustment on the algorithm provides a considerable efficiency. We used two different versions of this algorithm depending on the data size, cluster size and serial or parallel environment. We call these algorithms k-means-inbd and k-means-ibd. Furthermore, both serial and parallel implementations were implemented; MapReduce was used for this purpose. The results indicate that using this new enhanced algorithm not only provided a considerable performance improvement over the classic algorithm but also outperformed an improved k-means algorithm from the literature in all tested cases. When the number of the iterations increased, the amount of work k-means had to perform increases linearly. Also increasing the number of clusters resulted in higher number of iterations, hence, it resulted in a linear increase in workload for k-means. However, since the amount of work performed by the proposed algorithms tend to become smaller at each iteration (after the initial adjustment period), the total amount of work that needed to be performed compared to k-means kept decreasing.

Secondly, we present a new hybrid solution that improves clustering quality, detects outlier data points and eliminates Empty Clustering problem significantly. Our model has a runtime disadvantage with respect to original k-means; however, both of them are linear with respect to the total number of the data points. To test our model we used two data sets, one of which is very crowded and difficult to cluster when $k >> 1$. We compared our model with original k-means and several

other improved clustering solutions. The experiments show that it outperforms other models and shows significant clustering quality gain.

Thirdly, we propose a novel optimization framework to menu optimization. Optimization of ATM menu structures is an important problem to reduce the delay experienced by the users and for the efficiency of utilizing ATMs. It is also imperative to formulate the optimization problem in a generic fashion so that many ATM menu types can be addressed within a single optimization framework. In this study, we built a novel MIP framework to address this open question, which has never been systematically investigated in the literature before. We present the results of performance evaluations of our optimization model by using real data collected from a large transaction database. Our results reveal that significant reduction in click count, thereby, in ATM transaction completion time is possible when the menu structure is optimized by the novel MIP framework presented in this study.

6.1 Future Work

This research can be extended in several ways, such as:

- Mobile device version of this model can be studied. The developer of mobile device can include the library to its application source code and set its parameters. Then application periodically can connect to main optimizer which will be in the cloud and find optimum slide menu items or tab bar menu items for particular application based on customer usage.
- The optimization of large menus that is difficult to find with MIP model can be researched. Here, meta-heuristic methods can be applied to search the solution space.
- Menu optimization and its correlation with demographic studies is another topic to be researched.

REFERENCES

[1] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[2] Melissa Dawe. Understanding mobile phone requirements for young adults with cognitive disabilities. In *Proceedings of the 9th international ACM SIGACCESS conference on Computers and accessibility*, pages 179–186. ACM, 2007.

[3] Jakob Nielsen. *Usability engineering*. Elsevier, 1994.

[4] Kamonwan Taohai, Suphakant Phimoltares, and Nagul Cooharojananone. Usability comparisons of seven main functions for automated teller machine (atm) banking service of five banks in thailand. In *Computational Science and Its Applications (ICCSA), 2010 International Conference on*, pages 176–182. IEEE, 2010.

[5] Barry W Boehm, John R Brown, and Hans Kaspar. *Characteristics of software quality*. North-Holland, 1978.

[6] Luis Olsina and Gustavo Rossi. Measuring Web application quality with WebQEM. *IEEE Multimedia*, (4):20–29, 2002.

[7] J A McCall, P K Richards, and G F Walters. Factors in Software Quality, Voll. I, II, III: Final Tech. Report. Technical report, RADC-TR-77-369, Rome Air Development Center, Air Force System Command, Griffiss Air Force Base, NY, 1977.

[8] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. California, USA, 1967.

[9] Stavros G Kolliopoulos and Satish Rao. A nearly linear-time approximation scheme for the euclidean k-median problem. In *Algorithms-ESA'99*, pages 378–389. Springer, 1999.

[10] Sanjeev Arora, Prabhakar Raghavan, and Satish Rao. Approximation schemes for euclidean k-medians and related problems. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 106–113. ACM, 1998.

[11] Pankaj K Agarwal and Cecilia Magdalena Procopiuc. Exact and approximation algorithms for clustering. *Algorithmica*, 33(2):201–226, 2002.

[12] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Applied statistics*, pages 100–108, 1979.

[13] Xiaobo Li and Zhixi Fang. Parallel clustering algorithms. *Parallel Computing*, 11(3):275–290, 1989.

[14] Clark F Olson. Parallel algorithms for hierarchical clustering. *Parallel computing*, 21(8):1313–1325, 1995.

[15] Ralf Lämmel. Google's mapreduce programming model—revisited. *Science of computer programming*, 70(1):1–30, 2008.

[16] Tom White. *Hadoop: the definitive guide: the definitive guide.* " O'Reilly Media, Inc.", 2009.

[17] Yingyi Bu, Bill Howe, Magdalena Balazinska, and Michael D Ernst. Haloop: Efficient iterative data processing on large clusters. *Proceedings of the VLDB Endowment*, 3(1-2):285–296, 2010.

[18] Sean Owen, Robin Anil, Ted Dunning, and Ellen Friedman. *Mahout in action*. Manning Shelter Island, 2011.

[19] Rui Maximo Esteves, Rui Pais, and Chunming Rong. K-means clustering in the cloud–a mahout test. In *Advanced Information Networking and Applications (WAINA), 2011 IEEE Workshops of International Conference on*, pages 514–519. IEEE, 2011.

[20] Rui Máximo Esteves and Chunming Rong. Using mahout for clustering wikipedia's latest articles: a comparison between k-means and fuzzy c-means in the cloud. In *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*, pages 565–569. IEEE, 2011.

[21] Eric Lee and James MacGregor. Minimizing user search time in menu retrieval systems. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 27(2):157–162, 1985.

[22] Kent L Norman and College Park. Better Design of Menu Selection Systems Through Cognitive Psychology and Human Factors. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 50(3):556–559, 2015.

[23] Dwight P Miller. The depth/breadth tradeoff in hierarchical computer menus. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 25, pages 296–300. SAGE Publications, 1981.

[24] Kathleen Snowberry, Stanley R Parkinson, and Norwood Sisson. Computer display menus. *Ergonomics*, 26(7):699–712, 1983.

[25] Kent L Norman and John P Chin. The effect of tree structure on search in a hierarchical menu selection system. *Behaviour & Information Technology*, 7(1):51–65, 1988.

[26] Panayiotis G Zaphiris. Depth vs Breath in the Arrangement of Web Links. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 44, pages 453–456. SAGE Publications, 2000.

[27] Ian H Witten, John G Cleary, and Saul Greenberg. On frequency-based menu-splitting algorithms. *International Journal of Man-Machine Studies*, 21(2):135–148, 1984.

[28] Andrew Sears and Ben Shneiderman. Split menus: effectively using selection frequency to organize menus. *ACM Transactions on Computer-Human Interaction*, 1(1):27–51, 1994.

[29] Gregory Francis. Designing Multifunction Displays: An Optimization Approach. *International Journal of Cognitive Ergonomics*, 4(2):107–124, 2000.

[30] Harold Thimbleby. Analysis and Simulation of User Interfaces. In *People and Computers*, pages 221–237. Springer, 2000.

[31] K R Cave and J M Wolfe. Modeling the role of parallel processing in visual search. *Cognitive Psychology*, 22:225–271, 1990.

[32] Baili Liu, Gregory Francis, and Gavriel Salvendy. Applying models of visual search to menu design. *International Journal of Human Computer Studies*, 56(3):307–330, 2002.

[33] Luigi Troiano and Cosimo Birtolo. Genetic algorithms supporting generative design of user interfaces: Examples. *Information Sciences*, 259:433–451, 2014.

[34] Luigi Troiano, Cosimo Birtolo, and Roberto Armenise. Searching optimal menu layouts by linear genetic programming. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–18, 2015.

[35] Shouichi Matsui and Seiji Yamada. A genetic algorithm for optimizing hierarchical menus. *2008 IEEE Congress on Evolutionary Computation, CEC 2008*, pages 2851–2858, 2008.

[36] Gilles Bailly, Antti Oulasvirta, Timo Kötzing, and Sabrina Hoppe. MenuOptimizer: Interactive Optimization of Menu Systems. In *The 26th*

annual ACM symposium on User interface software and technology (ACM UIST), 2013.

[37] Mikhail V Goubko and Alexander I Danilenko. An Automated Routine for Menu Structure Optimization. In *Eics 2010: Proceedings of the 2010 Acm Sigchi Symposium on Engineering Interactive Computing Systems*, pages 67–76, 2010.

[38] Robert St Amant, Thomas E Horton, and Frank E Ritter. Model-based evaluation of expert cell phone menu interaction. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 14(1):1, 2007.

[39] Yusuke Fukazawa. Automatic Cell Phone Menu Customization Based on User Operation History. *Transactions of Japaneese Society on Artificial Intelligence*, 25(1):68–77, 2010.

[40] Barry Smyth and Paul Cotter. Intelligent navigation for mobile internet portals. In *IJCAI Workshop on AI Moves to IA: Workshop on Artificial Intelligence, Information Access, and Mobile Computing. The 18th International Joint Conference on Artificial Intelligence*. Citeseer, 2003.

[41] Ying Tan and Yuanchun Zhu. Fireworks algorithm for optimization. In *Advances in Swarm Intelligence*, pages 355–364. Springer, 2010.

[42] Shaoqiu Zheng, Andreas Janecek, and Ying Tan. Enhanced fireworks algorithm. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 2069–2077. IEEE, 2013.

[43] Hirakata Kobayashi. Automatic teller machine, 1986.

[44] K. Taohai, S. Phimoltares, and N. Cooharojananone. Usability Comparisons of Seven Main Functions for Automated Teller Machine (ATM) Banking Service of Five Banks in Thailand. *Computational Science and Its Applications (ICCSA), 2010 International Conference on*, pages 176–182, 2010.

[45] Nagul Cooharojananone, Kamonwan Taohai, and Suphakant Phimoltares. A new design of ATM interface for banking services in Thailand. *Proceedings - 2010 10th Annual International Symposium on Applications and the Internet, SAINT 2010*, pages 312–315, 2010.

[46] Ghulam Mujtaba and Tariq Mahmood. Adaptive Automated Teller Machines—Part I. In *Information and Communication Technologies (ICICT), 2011 International Conference on*, pages 1–6. IEEE, 2011.

[47] Kevin Curran and David King. Investigating the human computer interaction problems with automated teller machine navigation menus. *Interactive Technology and Smart Education*, 5(1):59–79, 2008.

[48] Tenzile Gul Apari, Fatma Molu, Nur Findik, and Mustafa Dalci. User Experience approach in financial services. *2013 The International Conference on Technological Advances in Electrical, Electronics and Computer Engineering, TAEECE 2013*, pages 400–403, 2013.

[49] Vera Hollink, Maarten van Someren, and Bob J Wielinga. Navigation behavior models for link structure optimization. *User Modeling and User-Adapted Interaction*, 17(4):339–377, 2007.

[50] Girish Krishnan, Sanjay Kumar, C. R. Jithin, Vinay V. Panicker, and R. Sridharan. Service innovation for the user interface of an ATM catering to the needs of the student community. In *Industrial Engineering and Engineering Management (IEEM), 2011 IEEE International Conference on*, pages 1180–1184. IEEE, 2011.

[51] Anita H M Cremers, Jacomien G M de Jong, and Johan S. Van Balken. User-centered design with illiterate persons: The case of the ATM user interface. *Lecture Notes in Computer Science*, 5105 LNCS:713–720, 2008.

[52] Indrani Medhi, Somani Patnaik, Emma Brunskill, S.N. Nagasena Gautama, William Thies, and Kentaro Toyama. Designing mobile interfaces for novice and low-literacy users. *ACM Transactions on Computer-Human Interaction*, 18(1):1–28, 2011.

[53] Andrew Thatcher, Farhaana Shaik, and Claus Zimmerman. Attitudes of semi-literate and literate bank account holders to the use of automatic teller machines (ATMs). *International Journal of Industrial Ergonomics*, 35(2 SPEC. ISS.):115–130, 2005.

[54] Mengxing Zhang, Feng Wang, Hui Deng, and Jibin Yin. A survey on human computer interaction technology for atm.

[55] Cheng Chu, Sang Kyun Kim, Yi-An Lin, YuanYuan Yu, Gary Bradski, Andrew Y Ng, and Kunle Olukotun. Map-reduce for machine learning on multicore. *Advances in neural information processing systems*, 19:281, 2007.

[56] Hung-chih Yang, Ali Dasdan, Ruey-Lung Hsiao, and D Stott Parker. Map-reduce-merge: simplified relational data processing on large clusters. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 1029–1040. ACM, 2007.

[57] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: a flexible data processing tool. *Communications of the ACM*, 53(1):72–77, 2010.

[58] Colby Ranger, Ramanan Raghuraman, Arun Penmetsa, Gary Bradski, and Christos Kozyrakis. Evaluating mapreduce for multi-core and multiprocessor systems. In *High Performance Computer Architecture, 2007. HPCA 2007. IEEE 13th International Symposium on*, pages 13–24. Ieee, 2007.

[59] Tyson Condie, Neil Conway, Peter Alvaro, Joseph M Hellerstein, Khaled Elmeleegy, and Russell Sears. Mapreduce online. In *NSDI*, volume 10, page 20, 2010.

[60] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[61] Weizhong Zhao, Huifang Ma, and Qing He. Parallel k-means clustering based on mapreduce. In *Cloud Computing*, pages 674–679. Springer, 2009.

[62] AM Fahim, AM Salem, FA Torkey, and MA Ramadan. An efficient enhanced k-means clustering algorithm. *Journal of Zhejiang University SCIENCE A*, 7(10):1626–1633, 2006.

[63] David Arthur and Sergei Vassilvitskii. How slow is the k-means method? In *Proceedings of the twenty-second annual symposium on Computational geometry*, pages 144–153. ACM, 2006.

[64] Donald E Knuth. Dynamic huffman coding. *Journal of algorithms*, 6(2):163–180, 1985.

[65] Charles Elkan. Using the triangle inequality to accelerate k-means. In *ICML*, volume 3, pages 147–153, 2003.

[66] Hadi Fanaee-T and Joao Gama. Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*, 2(2-3):113–127, 2014.

[67] Paul S Bradley and Usama M Fayyad. Refining initial points for k-means clustering. In *ICML*, volume 98, pages 91–99. Citeseer, 1998.

[68] PS Bradley, KP Bennett, and Ayhan Demiriz. Constrained k-means clustering. *Microsoft Research, Redmond*, pages 1–8, 2000.

[69] Yi Lu, Shiyong Lu, Farshad Fotouhi, Youping Deng, and Susan J Brown. Fgka: A fast genetic k-means clustering algorithm. In *Proceedings of the 2004 ACM symposium on Applied computing*, pages 622–623. ACM, 2004.

[70] CPLEX V12.1 User's Manual, 2009.

[71] Laurence A Wolsey. *Mixed integer programming*. Wiley Online Library, 2008.

[72] Jeyhun Karimov, M. Ozbayoglu, B. Tavli, and E. Dogdu. Generic menu optimization for multi-profile customer systems. In *Systems Engineering (ISSE), 2015 International Symposium on*. IEEE, 2015.

CURRICULUM VITAE

Personal Information

Surname, Name	: KARIMOV Jeyhun
Citizenship	: Azerbaijan
Date and Place of Birth	: 06.08.1988 Azerbaijan, Baku
Marital Status	: Not Married
Telephone	: (90) 507 0849234
e-mail	: je.karimov@gmail.com

Education

Level	**Training Unit**	**Graduation Date**
Master Degree (with scholarship)	TOBB University of Economics and Technology	2015
Bachelor Degree	Middle East Technical University	2012

Job Experience

Year	**Place**	**Duty**
2010-2011	Siemens EC	SDE
2012-2013	Mobicom S.A.	SDE
2014	Nivelsoft	SDE

Foreign Language

English (Advanced)
Russian (Advanced)

Publications

- Jeyhun Karimov , Murat Ozbayoglu, Bulent Tavli and Erdogan Dogdu,

"Generic Menu Optimization for Multi-profile Customer Systems". IEEE International Symposium on Systems Engineering, Rome, 2015.

- Jeyhun Karimov , Murat Ozbayoglu and Erdogan Dogdu, "k-means Performance Improvements with Centroid Calculation Heuristics both for Serial and Parallel environments". IEEE International Congress on Big Data , New York, 2015

- Jeyhun Karimov , and Murat Ozbayoglu , "High quality clustering of big data and solving empty-clustering problem with an evolutionary hybrid algorithm". IEEE International Conference on Big Data , Santa Clara, 2015

- Jeyhun Karimov , and Murat Ozbayoglu , "Clustering Quality Improvement of K-means Using a Hybrid Evolutionary Model". Complex Adaptive Systems, Santa Jose, 2015.