Computational Economics
https://doi.org/10.1007/s10614-021-10147-0

# Menu Optimization for Multi-Profile Customer Systems on Large Scale Data

Jeyhun Karimov[1] · Murat Ozbayoglu[2] · Bulent Tavli[2] · Erdogan Dogdu[3]

Accepted: 24 June 2021

**Abstract**
Everyday, a majority of the people, most probably several times, use the banking applications through online applications or physical ATM (Automated Teller Machine) devices for managing their financial transactions. However, most financial institutions provide static user interfaces regardless of the needs for different customers. Saving even a few seconds for each transaction through more personalized interface design might not only result in higher efficiency, but also result in customer satisfaction and increased market share among the competitors. In ATM Graphical User Interface (GUI) design, transaction completion time is, arguably, one of the most important metrics to quantify customer satisfaction. Optimizing GUI menu structures has been pursued and many heuristic techniques for this purpose are present. However, menu optimization by employing an exact mathematical optimization framework has never been performed in the literature. We cast the ATM menu optimization problem as a Mixed Integer Programming (MIP) framework. All the parameters of the MIP framework are derived from a comprehensive actual ATM menu usage database. We also proposed two heuristic approaches to reduce the computational complexity. Our solution can be accustomed with ergonomic factors and can easily be tailored for optimization of various menu design problems. Performance evaluations of our solutions by using actual ATM data reveal the superior performance of our optimization solution.

**Keywords** HCI design and evaluation methods · Mixed integer programming · Heuristic algorithms · Graphical user interfaces · Interaction design theory · Scalable data processing

Extended author information available on the last page of the article

## 1 Introduction

Graphical User Interface (GUI) is an important component of interactive computer applications with multiple user profiles (*i.e.*, users exhibiting different usage behaviors and patterns). Interactive web portals, Automated Teller Machines (ATMs), smart phone applications are among many of these applications. There is an opportunity to customize user interfaces, especially user menus, of these systems, for different user profiles based on the usage behaviours so that user satisfaction, service quality, and profitability can be increased. For example decreasing transaction completion time benefits many users of ATMs and the banks that own those ATMs. This problem, in general, is called "menu optimization" or "menu customization" problem Goubko and Danilenko (2010).

A naive glimpse to the menu optimization problem may render the problem, erroneously, as a trivial one. For example, determining the most clicked menu items and positioning them in top menu screens, is a quick solution Jain (2012). In fact, such a heuristic may reduce the average completion time provided that frequently used items are not already at the upper layers of menu hierarchy. However, there are several constraints that prevent such a heuristic to be implemented freely. For example, there is an upper limit on the number of items that can be placed in a single screen, due to screen space and/or the ergonomics of GUI design Al-Saleh and Bendak (2013). Besides those, menu items cannot be arbitrarily grouped under a single menu item because they may not be related to each other (semantics) Danilenko and Goubko (2013). Indeed, usability is one of the main factors affecting the quality of software systems McCall et al. (1977); Mayer et al. (2016). Therefore, in optimizing menus ensuring usability is vital. For example, the structure of the menu cannot contain any ambiguous placement of menu items. Hence, optimizing a menu structure necessitates taking many constraints into consideration which accordingly requires construction of a well defined mathematical optimization framework. Furthermore, abstractions mapping constraints arising due to ergonomics, semantics, resource limitations, and functional behaviors should be captured by concise and efficient mathematical representations to be used in the optimization framework.

Optimization of ATM menu design is a topic investigated in the literature from various perspectives and for maximizing different objectives Zhang et al. (2012). Our main contribution in this study is the creation of a generic mathematical optimization model which is guaranteed to uncover the optimum design among many possible alternatives under the given constraints within the defined variable space. We transformed the ATM menu optimization problem into a network flow problem by treating the menu items as vertices of the graph, links between the menu items as edges, and the users' navigation among the vertices as the flows. The objective is to position the menu items in such a way that the total weighted sum of flows on the vertices is the achievable minimum without violating the usability, ergonomics, and resource constraints. All parameters of the model are derived from a large database of actual ATM logs. Since our optimization model is a Mixed Integer Program (MIP), it is rather straightforward to extend its applicability to many GUI menu optimization problems other than the ATM menu optimization problem by customizing the parameters, constraints, and objective function. To the best of our knowledge, such optimization framework has never been constructed in the context of ATM menu optimization in the literature. One exception is an earlier study where the initial design and performance analysis of our MIP-based solution has been introduced Karimov et al. (2015b). In this study we improved our model and significantly expanded our performance evaluations. Nevertheless, in this study we seek answers to the following research questions, the answers of which also represent our novel contributions enumerated as follows:

1. How can we unearth the behaviors and actions of a huge number of users buried in ATM logs to create efficient abstractions and parameters to provide input to a mathematical programming framework?
2. How can we transform the ATM menu optimization problem into an equivalent network flow problem that can be modeled as an MIP model?
3. How can we embed the ergonomics, usability, and resource constraints into the MIP model?
4. Can we construct a heuristic model that has much lower complexity than the exact model?
5. What is the extent of the improvements brought by the MIP model in comparison to simple heuristics like moving up the highly accessed menu items to the upper layer menu screens?

The rest of the paper is organized as follows. Literature review is presented in Sect. 2. We present the system model in Sect. 3. Experiments are provided in Sect. 4. Conclusions are drawn and open questions for future research are given in Sect. 5.

## 2 Related Work

The selection and design of effective menus is at the core of human computer interaction Norman (1991). There has been significant attention to this issue previously. We organized the existing studies in literature in three groups. The first group of studies are focused on improving the performance of hierarchical menus. The second group of studies emphasize the use of heuristic approaches for menu optimization. The third group of studies specifically address ATM menu optimization.

### 2.1 Hierarchical Menus

Hierarchical menu optimization has been the main topic of many studies in literature Miller (1981), Witten et al. (1984), Lee and MacGregor (1985), Norman and Chin (1988), Francis (2000), Thimbleby (2000), Norman (2008). One of the earliest research studies conducted on menu optimization is Witten et al. (1984) that used the hierarchical index of a digital phone book using the access frequencies of phone numbers. Another pioneering study is Lee and MacGregor (1985) where an analytical model for search time in menus is proposed. In Francis (2000), a quantitative approach for hierarchical menu design is proposed. Using Huffman Coding to optimize the menu structure based on the probabilities of menu items' access times is another approach proposed in Thimbleby (2000). The advantage of menu's breadth over its depth is advocated in Norman (2008), furthermore, the importance of cognitive psychology and human factors research in the design of menu selection systems are emphasized. It is also shown experimentally that using broader and shallower menus instead of deeper and narrower ones make it easier and faster to access information Miller (1981). Context-awareness and adaptiveness can also be considered as influential factors in responsive, dynamic menu designs Ghiani et al. (2015). Another noteworthy approach is using hybrid menus (*i.e.*, menus with larger breadth at deeper layers) which are shown to be more efficient than menus that became narrower towards the end Norman and Chin (1988). Split menus are proposed in Sears and Shneiderman (1994) where frequently accessed menu items are located at the top of the menu groups or menu pages by splitting the menus.

## 2.2 Heuristic Approaches

Solving an exact mathematical programming model is, generally, not feasible or impossible due to the prohibitive computational complexity. Therefore, obtaining good enough solutions by using heuristic or approximate algorithms is preferable in practice. There is a wealth of literature that focus on using evolutionary algorithms and heuristics for menu optimization Cave and Wolfe (1990), Liu et al. (2002), Smyth and Cotter (2003), Amant et al. (2007), Hollink et al. (2007), Matsui and Yamada (2008), Fukazawa et al. (2010), Goubko and Danilenko (2010), Bailly et al. (2013), Troiano and Birtolo (2014), Troiano et al. (2016). Guided-Search algorithm is proposed in Cave and Wolfe (1990) for defining the necessary components of a good user interface. In Liu et al. (2002), Guided-Search algorithm's performance is investigated under different scenarios in order to increase the satisfaction level. Genetic algorithms were used for menu structure optimization and color scheme selection in Troiano and Birtolo (2014), Troiano et al. (2016). In Matsui and Yamada (2008), genetic algorithm in conjunction to simulated annealing is employed for optimizing the performance of menus on cell phones. It is argued that automated menu optimization and design is desirable but semantics always play a role, therefore, utilization of human assistance as part of the optimization and design process is suggested. For example, a menu design tool that employs interactive optimization approach to menu optimization is proposed in Bailly et al. (2013). The tool allows designers to choose good solutions and group items while delegating computational problems to an ant colony optimizer. In Goubko and Danilenko (2010), the use of informal judgements for the final menu structure along with an automated procedure is proposed. In Amant et al. (2007), GUI optimization techniques for evaluating and improving cell phone usability with efficient hierarchical menu design is investigated. In Fukazawa et al. (2010), menu option usage frequencies and recent usage history data are employed as inputs to Ranking SVM method for determining the best cell phone menu layout. In Hollink et al. (2007), interactive web site menu optimization is performed with hill-climbing method which minimizes average time to reach target pages.

### 2.3 ATM Menu Optimization

There have been several studies to optimize themenu structure on ATMs Kobayashi (1986), Thatcher et al. (2005), Curran and King (2008), Cremers et al. (2008), Taohai et al. (2010), Cooharojananone et al. (2010), Krishnan et al. (2011), Zhang et al. (2012) for improved customer satisfaction. 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. Such an objective is also employed for other types of menus (*e.g.*, online banking user interface is optimized with the same objective in Apari et al. (2013)). In Zhang et al. (2012), literature on Human Computer Interaction Technology for ATMs is reviewed. In certain studies on ATM menu optimization, specific assumptions about users are made and menu optimization is performed under these assumptions Cremers et al. (2008), Krishnan et al. (2011). In Krishnan et al. (2011), authors proposed solutions for optimizing ATM menus for the use of student communities. In Cremers et al. (2008), optimized ATM menus are designed for illiterate people and these menus are tested on functionally illiterate users. Designs of different ATM menus with speech-based and icon-based interfaces for literate, semi-literate, or elderly people are studied earlier in Thatcher et al. (2005); Huang et al. (2019). Note that GUI optimization for novice and low-literate customer use is also an active research area in various contexts other than ATM menu optimization Nielsen (1994), Dawe (2007), Medhi et al. (2011), Chanco et al. (2019).

Nevertheless, to the best of our knowledge, our menu optimization solution is the first mathematical programming based study on menu optimization which guarantees the optimum solution. Furthermore, our optimization framework is not designed for a specific group of people or a specific use case. Instead, our formulations are rather generic and can be applied to many menu optimization contexts with minor modifications.

## 3 System Model

In Subsect. 3.1 we present the problem definition to form the foundation for the details of the optimization framework described in Subsect. 3.2.

### 3.1 Problem Definition

The specific problem we would like to solve in this study is to optimize the structure of an ATM menu. As in most optimization problems, also in our case there are several constituents of the system to be optimized which are itemized as follows:

– Transforming the ATM menu optimization problem into a suitable mathematical model is the starting point. We decide to model an ATM menu as a graph $G(V, E)$ where $E$ and $V$ are the edges and vertices of the graph, respectively. A sample ATM menu illustrated as a tree is presented in Fig. 1. Note that the tree

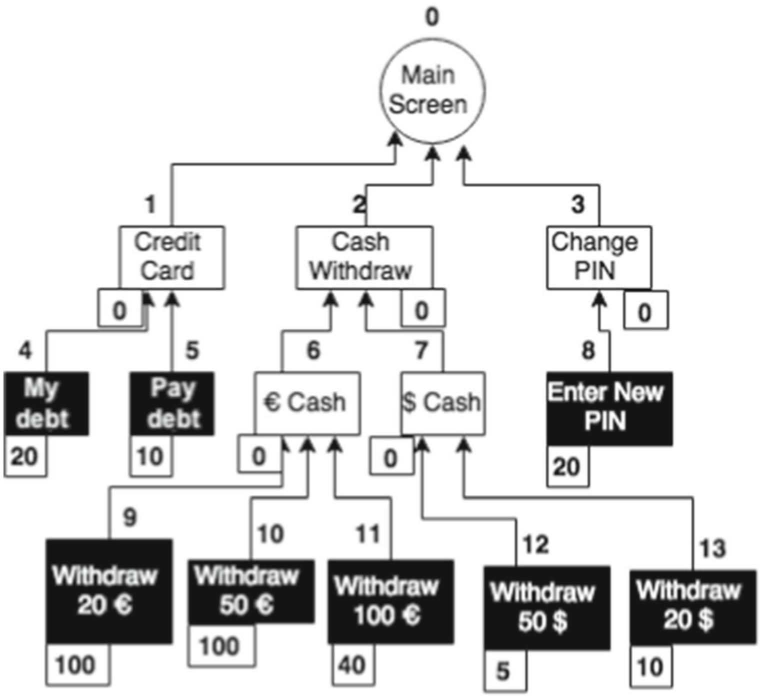


**Fig. 1** A sample menu structure of ATM before optimization. Rectangles represent menu items, the numbers above boxes represent their ID, the ones below boxes represent click counts for the particular menu item and black boxes represent leaf menu items (*i.e.*, terminal menu items in the click path)

in the figure is intentionally depicted as simple as possible for the sake of clarity. In the tree, menu items are the vertices of the graph and links between the menu items are the edges. Leaf nodes are represented by black boxes where actual transactions take place meanwhile boxes with white background are intermediary nodes used for transiting from one menu screen to another.

- The graph problem can further be transformed into a network flow problem. Leaf nodes are the source nodes while the disc shaped initial screen is the destination for all source nodes. The flow injected through the source nodes represent the amount of transactions destined for the source node itself which is bound to terminate at the sink node (*i.e.*, initial screen). Note that in practice actual flow is from the initial screen to the leaf nodes, however, problem construction is more efficient by reversing the flow, yet, reversing the flow direction (*i.e.*, reversing the ingress and egress nodes) does not interfere with the optimization objective.
- The logs of user transactions have to be mined and for each user its corresponding customer profile must be determined. The click count of each menu item is not directly determined as the number of clicks for a particular menu item from the user logs. Instead, our objective is to get the actual aim of the user. The performed user clicks at the intermediate nodes on the path to the

desired menu item do not contribute to the click count of the leaf node or any node that is on its path. The only menu item that gets the credit from the click is the one that the user is actually trying to reach. For example, considering Fig. 1 if a particular user traverses the path [Main screen]→[Cash withdraw]→[€ Cash]→[Withdraw 50 €] then his final objective is to reach the [Withdraw 50 €] menu item (*i.e.*, his aim is not to reach [Cash withdraw] or [€ Cash] menu items, *per se*). So even though the user clicks [Cash withdraw] or [€ Cash] menu items (in order to reach [Withdraw 50 €] ), the click counts of the aforementioned menu items do not get credited. Therefore, it is possible that a certain menu item under another menu item can have a higher click count than its parent menu item. Note that mining, clustering, analysis, and classification of ATM usage data has been performed in our earlier studies (Karimov and Ozbayoglu,2015; Karimov et al.,2015a), hence, in this paper we do not go into the details of data processing and assume that the user logs are mined in an efficient way (*i.e.*, they are clustered and click counts of menu items for each cluster is known).

- Original graph representation of the ATM menu to be optimized is readily available (*i.e.*, actual menu structure in use by the financial establishment serving the customers). In other words, we start our optimization from the present ATM menu represented as a tree. We use the click counts obtained from the data logs as the amount of flow entering the network from each leaf node.
- The objective function is minimization of the transaction completion time which can be obtained by the sum of weighted flows on the vertices of the graph. For example, considering Fig. 1 repositioning the [€ Cash] menu item directly under the [Main Screen] will reduce the average transaction completion time (*i.e.*, reduce the total weighted flow).
- Ergonomic factors to be considered in the menu design should be taken into account in the form of mathematical representations which will be used as the constraints of the optimization model. Certain menus should not have more than a predefined maximum number of menu items on each menu screen (*e.g.*, in smart phone or ATM menus the space constraints are very restrictive). Our model handles this issue by limiting the number of children of each menu item. For example, we assume that the menu shown in Fig. 1 can have at most 3 children per node.
- Semantics is another factor to be carefully embodied in the optimization framework. Assume that we use Huffman coding Knuth (1985) to determine the positioning of all menu items so that the most clicked menu items are placed in top screens and the rest of the tree is reconstructed in this manner. Such an approach would give the best solution, however, considering Fig. 1, menu items [Enter New PIN] and [My Debt] could be placed under the same menu screen whereas menu items [Withdraw 20 $] and [Pay Debt] can be placed under another menu item. Such an arrangement at the same level would seriously hamper the usability of the reconstructed menu because of the ambiguity created. Therefore, at each menu screen there can, at most, be one generic menu item (*i.e.*, [Other Operations]).

The optimized version of the original menu is presented in Fig. 2. Although, in this particular case, determining the optimal menu for Fig. 1 is not extremely challenging, in practical menus brute force optimization is prohibitively time consuming.

## 3.2 Optimization Framework

Having presented the high level overview of our optimization approach, we will describe three optimization models. The first model is the Fully Optimum Menu (FOM) model which is an MIP model. The second model is the Scalable Optimum Menu (SOM) model which is a heuristic approach with lower computational complexity than the FOM model. The third model is the Quick Optimum Menu (QOM) model which is the model with lowest computational complexity, however, the price paid for the speed is the higher optimality gap.

### 3.2.1 FOM

The FOM model, or Fully Optimum Model, which is an MIP model, can be formulated as follows:

$$a_{ij} \in \{0, 1\} \ \forall i \in S_U, \forall j \in OUT_i \quad (1)$$

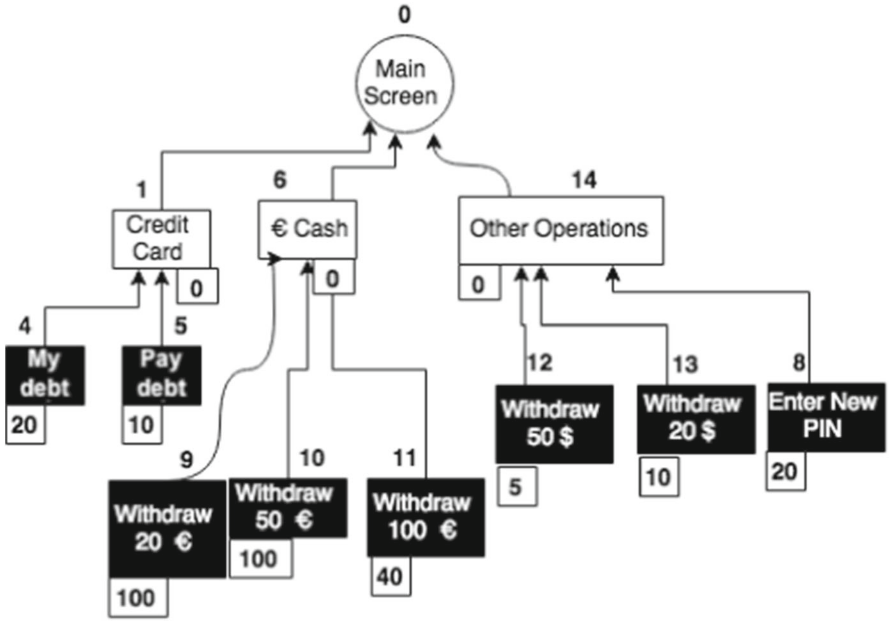


Fig. 2 A sample menu structure of ATM after optimization

$$f_{ij} \geq 0 \; \forall i \in S_U, \forall j \in OUT_i \tag{2}$$

$$f_{ij} \leq Ma_{ij}, \forall i \in S_U, j \in OUT_i \tag{3}$$

In this model, there are three types of nodes (vertices): original menu items (set $S_{ORG}$), set of combiner nodes (set $S_{COMB}$), and set of optimizer nodes (set $S_{OPT}$). The universal set of all nodes is represented by set $S_U$. The variables of the MIP model are $f_{ij}$ (flow on arc-($i$, $j$)) and $a_{ij}$ (binary variable indicating whether any flow exists on arc-($i$, $j$)). The constraints on the flows are presented in Equation 1 and Equation 2. The set $OUT_i$ represents the arcs going out of node-$i$. Equation 3 is used to establish the relation between the flows ($f_{ij}$) and their indicator variables ($a_{ij}$) through the use of $M$ which is a big number. Note that the Big $M$ technique is a well known technique in Integer Programming Wolsey (1998).

In Fig. 3 a sample network is illustrated which is useful in explaining the concepts of the FOM model. The difference between Fig. 1 (the original tree structure of the menu) and Fig. 3 is that in Fig. 3 two hypothetical node types are introduced: optimizers (triangle-like boxes) and combiners (romb-like boxes). Furthermore, new arcs are created (dashed lines) to integrate these nodes into the tree.

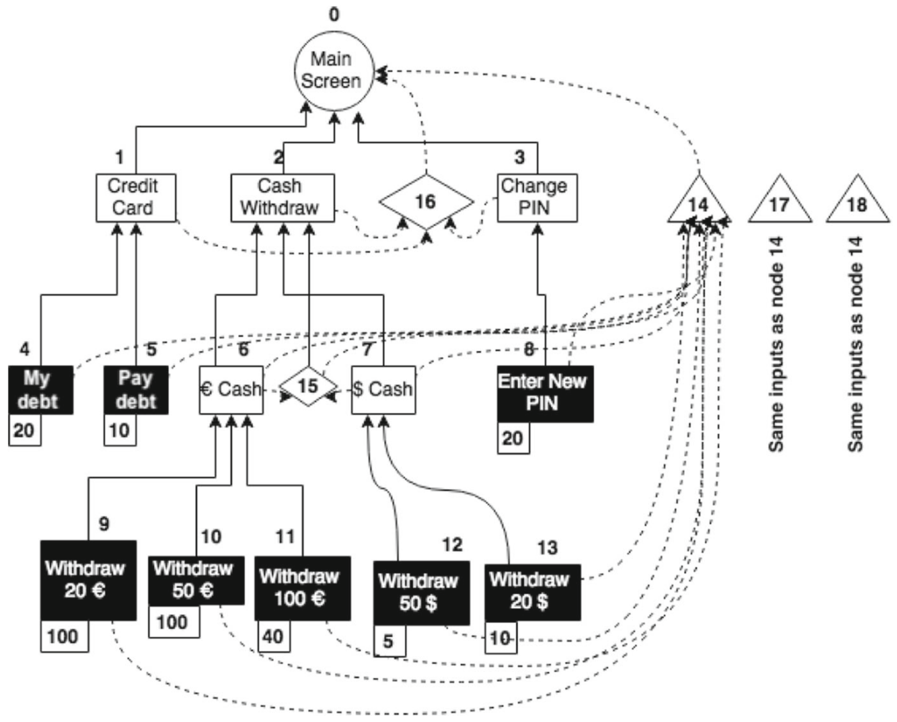


**Fig. 3** A sample menu structure with the introduction of combiners and optimizers

Optimizer nodes are used to bring the best leaf node or menu subtree to upper levels. Optimizer nodes have connections to all grandchildren nodes of their parent nodes, however, $a_{ij} \neq 0$ only for one of these connections (*i.e.*, only one of the nodes are relocated to the upper layer in the tree hierarchy). For example, in Fig. 3, node-14, node-17, and node-18 are optimizers and all lower level nodes with the same parent (Main Screen) are connected to them. The optimizer enables the selection of the best node to relocate to the upper layer as there can be only one child for each of the optimizers. In all levels of the tree hierarchy there are optimizers except at the leaf node level. In order not to make the illustration in Fig. 3 overcrowded, we display optimizers only in the first level. Optimizers can only connect to the same parent menu nodes with itself (*i.e.*, not to lower level menu nodes). The cost of connecting to an optimizer is zero. Nevertheless, if one of the down links of an optimizer node has non-zero flow, then the corresponding menu item at the other end of the link is moved up in the menu hierarchy.

Combiner nodes are used to combine same level nodes under one menu node and take them one level below. For example, in Fig. 3 node-6 is a combiner node. All the same level nodes are possible child nodes of the combiner. If some nodes in the same level are chosen to go downwards (so that the space necessary for possible upcoming nodes are made available), they are collected under the combiner node with a generic name (*e.g.*, Others) and their previous places are taken by other menu nodes from lower levels through the mechanisms provided by optimizer nodes.

The amount of flow injected by each node-$i$ is denoted as $C_i$ which is a parameter obtained from the user logs. For the sink node (Main Screen denoted as $S_0$) the total amount of incoming flow is the sum of the data injected at all other nodes as stated in Equation 4. Note that there is no outgoing flow from the sink node. For all other nodes the difference between the incoming and outgoing flows is equal to the injected flow at the particular node. The set $IN_i$ represents the arcs coming into node-$i$.

$$\sum_{j \in OUT_i} f_{ij} - \sum_{j \in IN_i} f_{ji} = \left\{ \begin{array}{c} -\sum_{i \in S-S_0} C_i, i = S_0 \\ C_i, \forall i \in S_U - S_0 \end{array} \right\} \tag{4}$$

Equation 5 states the constraint on the outgoing arcs from nodes. For optimizer nodes and combiner nodes there can at most be one outgoing arc. However, it is possible that there are no outgoing arcs from such nodes which implies that such a node is not needed for the optimized menu structure. For all other nodes there is exactly $L^i_{OUT}$ number of arcs going out of node-$i$ because a regular node must be reachable from the Main Screen even if there is no data injected by the particular node. Note that we set $L^i_{OUT} = 1$.

$$\sum_{j \in OUT_i} a_{ij} \left\{ \begin{array}{c} \leq 1, \forall i \in S_{OPT} \cup S_{COMB} \\ = L^i_{OUT}, \forall i \in S_{ORG} \end{array} \right\} \tag{5}$$

Equation 6 defines the constraint on incoming arcs to nodes. Optimizer nodes can have at most one incoming arc. For other nodes the limit on the number of incoming arcs is $L_{IN}^{i}$. Actually, $L_{IN}^{i}$ is the maximum allowable number of menu items in a menu screen.

$$\sum_{j \in IN_i} a_{ji} \left\{ \begin{array}{c} \leq 1, \forall i \in S_{OPT} \\ \leq L_{IN}^{i}, \text{otherwise} \end{array} \right\} \tag{6}$$

Weights of the arcs are given in Equation 7. All arcs except the ones incoming to optimizer nodes have the weight of one. Arcs leading directly to the optimizer nodes have zero weight because optimizer nodes are purely hypothetical nodes (*i.e.*, they will not appear in the final menu) and their sole role is to relocate certain nodes closer to the Main Screen. By zeroing the arc cost for such nodes we enable the cost-free upward relocation of selected nodes.

$$d_{ij} = \left\{ \begin{array}{c} 0, \text{if } j \in S_{OPT} \\ 1, \text{otherwise} \end{array} \right\} \forall i \in S_U, \forall j \in OUT_i \tag{7}$$

The objective function is given in Equation 8 which is the minimization of the weighted sum of flows over all arcs. If the total flow is minimized then the average transaction completion time is also minimized.

$$\text{Minimize} \sum_{i \in S_U} \sum_{j \in OUT_i} d_{ij} f_{ij} \tag{8}$$

In essence, FOM is a network flow problem. Click counts of the menu items and the initial menu tree are the main inputs of the model. By minimizing the flow in the tree we reduce the average time spent by the users for their ATM transactions. The solution of the FOM model will result in determining the optimum menu structure.

### 3.2.2 SOM

A more scalable version of FOM is SOM. Indeed, the main design philosophy of the SOM model is to decompose the menu tree into smaller pieces and run FOM for each smaller piece and recombine the solutions to obtain the final optimized menu tree. However, SOM does not guarantee finding the optimal solution, yet, computational complexity of SOM is lower than FOM. Therefore, SOM can work on much larger menus when compared to FOM. One of the key parameters that SOM takes as an input is the maximum number of menu items that can be contained in a decomposed subtree (*i.e.*, $C^M$). The original menu is decomposed into subtrees such that node count of each subtree cannot be greater than $C^M$. Then, FOM is run for all of them in parallel and results are combined.

**Algorithm 1:** Scalable Optimum Menu (SOM) Generation

```
Input: $MT$ (the menu tree) and $C^M$ (maximum size of a subtree to be processed
    by FOM)
Output: Computed tree structure
1  Let $desc$ be a field of nodes in $MT$ such that for arbitrary menu node-$n$ in $MT$,
   $n.desc$ shows count of all nodes under $n$'s subtree
2  Algorithm FindDesc($Node$-$n$)
3      if $n$ is a leaf node then
4          $n.desc = 0$
5      else
6          $n.desc = \sum_{c \in n.children} (FindDesc(c) + 1)$
7      end
8  Run FindDesc with head node of $MT$
9  Set $Q \leftarrow$ all leaf nodes of $MT$
10 forall $Nodes$ in $Q$ do
11     $t \leftarrow$ random Node from $MT$
12     while $t.parent \neq headNode$ & $t.parent.desc \leq C^M$ do
13         $t \leftarrow t.parent$
14     end
15     Remove ($C^M$ - t.parent.$desc$ ) leaf nodes from $t$'s subtree with minimum click
         counts and remove them also from $MT$
16     Update t.$desc$ and its parents' $desc$ fields
17     Save t's subtree: $checkpoint^t \leftarrow$ t's subtree
18     $MT \leftarrow MT - checkpoint^t$ remove t's subtree from MT
19     $suboptimal^t \leftarrow FOM(checkpoint^t)$. Send saved subtree to FOM to be
         optimized
20     Make $suboptimal^t$ a leaf node of $MT$ with $suboptimal^t.clickCount \leftarrow$ average
         of all nodes' click counts under $checkpoint^t$'s subtree
21     $MT \leftarrow MT \cup suboptimal^t$
22 end
23 Substitute saved leaf nodes with optimized subtrees returned from $FOM$ method
```

Algorithm 1 presents the pseudo code of SOM model. It takes the menu tree, *MT*, and $C^M$. First, FindDesc method needs to run to preprocess the menu tree. It computes the number of descendants under a particular menu node. In line 9, we put all leaf nodes of *MT* to a set *Q*. In lines 11-21 of the algorithm, we get a random leaf node (*t*) from the set *MT*. Then we look for its parent's child count (*desc* field). If the count is less than $C^M$ value, it means that we can go to the upper level. We stop when the number of nodes under a particular node is higher than $C^M$ value. After that, the algorithm removes $(C^M - n.desc)$ number of least clicked leaf nodes from node-*t*' subtree because least clicked nodes are not going to replace their location, so we can confidently remove them if needed. Then, we save the subtree under *t* as $checkpoint^t$ to be optimized by *FOM* model. The algorithm terminates successfully after the results obtained from *FOM* method are put in place of the previously replaced menu nodes.

Figure 4 illustrates the overall intuition of SOM method. As can be observed from the figure, we divide the original menu tree into subtrees with at most $C^M$ nodes and select the head nodes of the subtrees as representative nodes. For example, $n_1$ is the first representative of the lower right subtree within the dashed

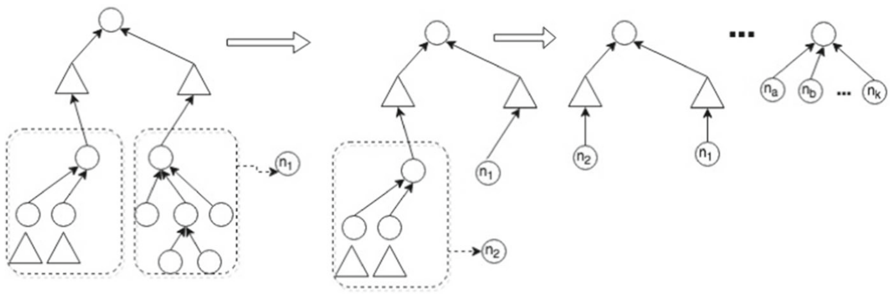


**Fig. 4** Overall intuition of SOM method

bounding enclosure (in the leftmost panel). Then, the lower left subtree is replaced by $n_2$. We continue in this manner until we reach the head node of the tree.

### 3.2.3 QOM

As stated in Sect. 1, employing greedy heuristics for menu optimization is possible, however, we advocated that such an approach will not result in the optimal solution, yet, such an assertion should also be quantitatively evaluated. Hence, we design the QOM model which is a fast and greedy approach. The overall procedure is given in Algorithm 2 where we first traverse all tree levels. In each level we find the most clicked menu nodes and if necessary move them upwards. If the maximum node size is exceeded under a particular menu screen then we select the least clicked nodes and push them downwards under *Others* node.

**Algorithm 2:** Quick Optimized Menu (QOM) Generation

```
   Input: $MT$ the menu tree
   Output: Computed tree structure
1  Let $H$ be a set keeping the most clicked nodes
2  Traverse $MT$ in depth first manner such that, every node keeps reference to most
     clicked node under its subtree
3  forall $Level\ l\ in\ MT$ do
4      forall $Node\ n\ in\ l$ do
5          $MT \leftarrow$ most clicked menu node under $n$'s subtree which is not element of
             $H$
6          $H \leftarrow H \cup MT$
7          $C \leftarrow$ most clicked menu node among $n$ direct children nodes
8          if $MT$ click count is greater than $C$ then
9              Add $MT$ to $n^{th}$ direct children nodes.
10         end
11         if $n.childCount > maxAllowedSize$ then
12             Select 2 least clicked menu nodes among children nodes of $n$, say $L_1$
                 and $L_2$
13             Add new menu node under $n$, named $Others$
14             Move $L_1$ and $L_2$ under $Others$ node
15         end
16     end
17 end
```

### 3.2.4 Completixy Analysis

FOM is the model which uses MIP to find the globally optimum menu. The optimum solution in FOM is guaranteed, however it comes with a computational cost. MIP solutions for optimization problems are known to be NP-hard Wolsey (2008). Since the problem is a node sequencing problem in a tree structure, theoretically it is possible to have a computational complexity up to O($n!$) representing the number of different node permutations where n refers to the number of nodes in the menu. Hence, FOM is the most costly algorithm compared to SOM and QOM.

For SOM, we have a model that depends on a trade-off between achieving global optimum and speed. The algorithm generates subtrees that consists of maximum $k$ nodes, which have optimum local node formations, but the subtree sequencing may not be optimum. Analysis of Algorithm 1 results in a complexity of O($n/k * k!nlog(n)$) where $k$ refers to the subtree size and $n$ refers to the number of nodes overall.

QOM is the fastest of the three models in our paper, since achieving high speed was the aim in this particular case. A greedy approach is adapted and as Algorithm 2 suggests the nodes are chosen sequentially with a depth-first-search like heuristic that has two inner loops which can theoretically have O($n$) nodes in each level. As a result a computational complexity of O($n^2$) can be achieved in the worst case.

Figure 7 illustrates the comparison of the computational times of the three models across different numbers of nodes. The results closely match with the theoretical complexity analyses of the three models.

## 4 Experiments

We used two data sets for the performance evaluations of our solutions. First data (DS1) set is an ATM log data set accumulated over 18 months in ATMs of a large financial institution. The data set consisted of 50 million ATM transactions from 160K unique customers using 2000 ATM machines. In this data set there are at most forty menu items. The second data set (DS2) is more comprehensive and the node count is as high as 400 nodes. Experiments are run on an Intel i7 (8 Core) machine with 32GB RAM. We used IBM Ilog Cplex for the numerical solutions of the optimization problems. Data mining, analysis, and user clustering for the data sets have been performed in our previous publications Karimov et al. (2015a); Karimov and Ozbayoglu (2015), hence, for details of data analysis we refer the reader to Karimov et al. (2015a); Karimov and Ozbayoglu (2015) and focus on the performance evaluations of the optimization of the menu structure.

To determine the performance gains brought by the MIP model (FOM), we first perform analysis by using DS1. Grouping the users according to their usage patterns into multiple clusters and creating an optimized menu for each cluster reduces the average transaction completion time. However, even when all users are treated as a homogeneous group and a single optimized menu is used, FOM still reduces the average transaction completion time from 117 s to 95 s (18.8% reduction in transaction completion time). When we grouped the users into three clusters (C1, C2, and C3 clusters) the decrease in transaction completion times are 37.9%, 32.8%, and 4.1% for C1, C2, and C3 clusters, respectively.

The following behavioral characteristics were observed in each cluster. C1 includes users that mostly took cash from ATM and looked into their balance. C2 includes users that mostly took cash from ATM and almost never looked into their balance, and C3 includes users that performed advanced operations in ATM such as money transfer, credit card operations, automatic bill payment, etc. The majority of the customers (approx. 60%) were in C1, approximately 30% were in C2 and approximately 10% were in C3 indicating most of the customers were interested in cash withdrawal transactions.

There is a significant reduction of clicks in C1 and C2 clusters because C1 and C2 clusters are composed of more homogeneous profiles than C3 cluster. Thus, grouping users into clusters and creating optimized menus for each cluster results in higher performance gains. In Table 1, percentages of users in each cluster benefiting or not benefiting from the menu optimization are presented. Although the lowest benefited cluster is C3, the highest benefited group is not C1. However, the highest reduction of transaction completion is achieved in C1 because the benefited group in C1 achieved more drastic reduction than the benefited group in C2.

To characterize the extent of the benefit of clustering, we obtained the reduction of transaction completion time as a function of number clusters which is presented

**Table 1** Users benefiting from menu optimization

| | Benefitted from optimization | Not benefitted from optimization |
|---|---|---|
| C1 cluster | 69% | 31% |
| C2 cluster | 78% | 22% |
| C3 cluster | 64% | 36% |

in Fig. 5. As the number of clusters increases the reduction in transaction completion time increases. However, rate of decrease gets lower as the number of clusters increase. Even though clustering improves the performance, aggressive clustering does not bring significant benefits.

To gain more insight into the mechanics of the FOM model we analyzed the transaction completion time reduction in the most frequently used items (*i.e.*, leaf nodes in the tree). Figure 6 shows the reduction for the top nine nodes. The reduction in the higher frequency nodes is larger than the lower frequency nodes because the impact of more frequently used nodes is higher on average transaction completion time.

In the rest of this section we will be using DS2 data set. Figure 7 presents the run times of FOM, SOM, and QOM models. Here, SOM($n$) shows max number of nodes within a block. Depending on the available computing power, one may chose different upper bounds on block sizes (nodes within triangles in Fig. 4). Due to the large differences in run times we opted to present the run times in log-scale (y-axis values are the $log_2$ values of the run times). As expected, FOM run time increases exponentially as the number of nodes in the menu tree increases. Note that the variations in the node count are achieved by selecting random subsets of leaf nodes. The rate of increase in running times of SOM is much lower than FOM due to the scalability brought by the decomposition of the menu tree. Furthermore, by taking advantage of parallelization, SOM runtime could further be reduced. QOM runtimes

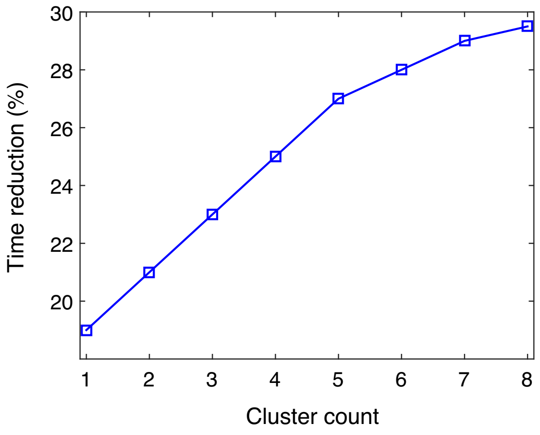


**Fig. 5** Time reduction vs. Cluster count

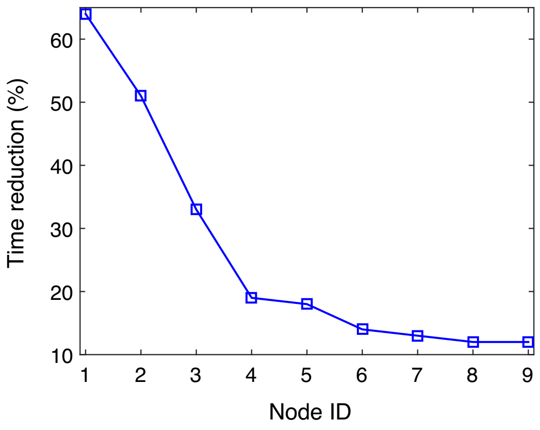


**Fig. 6** Time reduction vs. Node ID

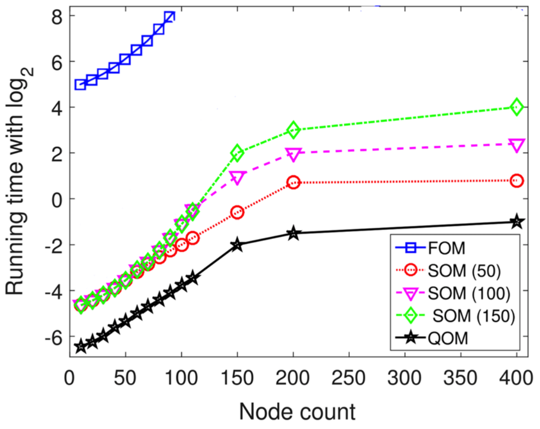


**Fig. 7** Effects of increasing node counts in menu on runtime

are the lowest among all (*i.e.*, several orders of magnitude lower than SOM) because of the simplicity of the model. However, the performance of the solutions produced by QOM is very much dependent on the initial tree structure due to its greedy design. From Fig. 7, it can easily be observed that SOM($n$) performs better in terms of running time once the node count exceeds $n$, as SOM($n$) benefits from parallelization of blocks.

In Fig. 8 we present the increase in the average transaction completion time for SOM and QOM in comparison to the optimal solution (*i.e.*, FOM). QOM performance is the worst among all models due to its dependency on initial menu structure and its greediness. For example, a value of 20% in Figure 8indicates that average transaction completion time is 5 times higher compared to the optimal FOM solution. The quality of SOM($n$) decreases as the node count gets larger than $n$. The reason for such behavior is that the number of sub-optimal solutions increase as the node count exceeds $n$ and the overall solution moves farther away from the global optimum.

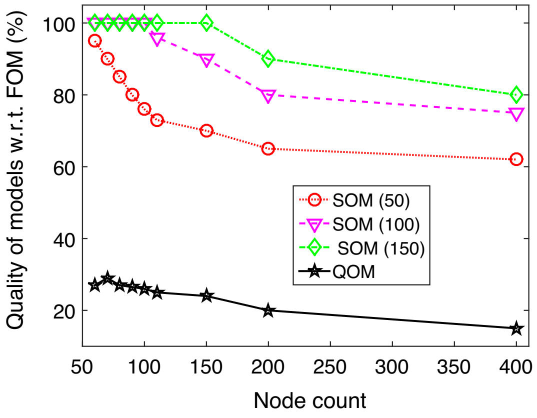


**Fig. 8** Dependency of quality of SOM and QOM models w.r.t. optimum (FOM) with increasing number of nodes

The effects of changing the maximum number of children of a node (*i.e.*, $L_{IN}^{i}$) on the objective function is presented in Fig. 9. As the number of possible children of a particular menu node gets larger the objective functions decrease for all models because of the increased flexibility to move up more menu items in the tree hierarchy. However, the rate of decrease gets lower as child count increases due to the diminishing marginal utilities principle. Here only SOM(100) model is presented for simplicity as we recognized there is no direct relation between SOM model's size and menu node's child size.

## 5 Conclusion and Future Research

In this study, we investigate GUI menu optimization problem in ATMs. We propose an MIP model (*i.e.*, FOM) for the optimization of the menu structure which guarantees the optimal solution. However, the scalability of FOM is limited, therefore, we construct a scalable suboptimal algorithm (*i.e.*, SOM) by decomposing the problem. We also proposed a greedy heuristic algorithm (*i.e.*, QOM). Since the

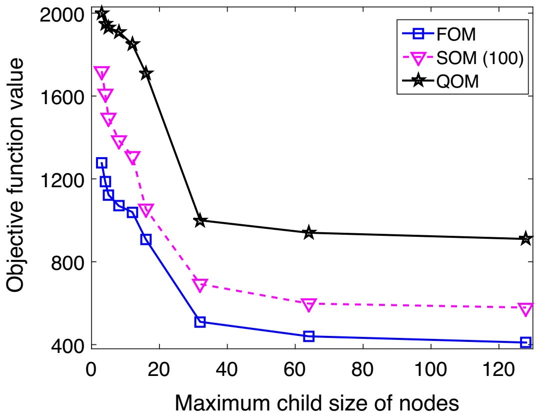


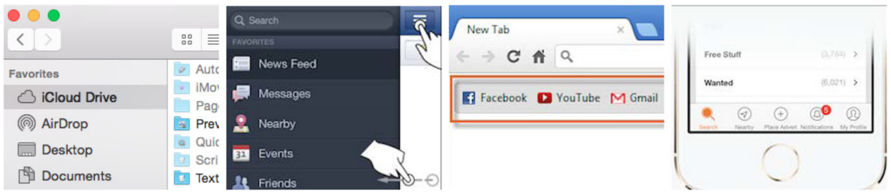
**Fig. 9** Effect of changing number of child nodes on objective function. Overall node count is 400

motivation for this paper is provided in the form of a series of research questions in Section I, we present conclusions in reply to these questions itemized as follows:

1. We used the utilization frequency of each menu item in constructing the MIP framework. Furthermore, we grouped the users into clusters for better optimization of cluster specific menus.
2. We transformed an ATM menu structure into an equivalent graph representation where menu items are the vertices and links between menu items are the edges of the graph. Furthermore, we used the menu item utilization as the source flows which are terminating at the Main Screen. Two novel node types are introduced to restructure the menu. Optimizer nodes and combiner nodes are used to enable the repositioning of the menu items upwards and downwards, respectively.
3. Ergonomics and usability factors are embedded into the MIP model in the form of constraints. To account for the maximum number of menu items that can be displayed in an ATM screen we limit the maximum number of children of each node. Moreover, with the constraints we ensure that user will not get ambiguous menu items related with parent menu.
4. We designed two suboptimal heuristic solutions for the reduction of the computational complexity of MIP model (FOM). SOM model is obtained by decomposition of the FOM model.
5. QOM model is a greedy local search algorithm which actually relocates the more accessed items upwards. Computational complexity of QOM is much lower than FOM, however, its performance is also significantly lower than FOM.

Our menu optimization solution is not applicable to ATM menus only. Indeed, many other menu types can also be optimized by using our framework with minor modifications. Interactive web pages and mobile device menus are among the possible application areas for our menu optimization framework as presented in Fig. 10.

In our application scenario, the number of menu items is not very high (*i.e.*, at most 400), therefore, we can obtain the optimal solution in reasonable time. Furthermore, menu optimization in our case is not done in real-time because there is no such requirement (*i.e.*, ATM menus are not frequently updated). However, for other menu optimization domains, much larger sets of menu items should be optimized (*e.g.*, web pages) where the use of SOM model is very much needed. In



**(a)** Mac OS **(b)** Slide menu **(c)** Web browser **(d)** Mobile tab bar

**Fig. 10** Some possible application domains of the proposed model

case of real-time menu optimization, QOM model can be used due to its lower computational complexity.

As mentioned, this paper concentrates mainly on the optimization part of generating optimal profile based user menus and the modules related with data mining are described in Karimov and Ozbayoglu (2015), Karimov et al. (2015a). Converting this system to incremental framework is an open direction for research. That is, currently we support optimization of profile based user menus with periodic batch jobs which computes all the data from scratch once the job fires. However, the data between different periodic jobs can be overlapped with some degree. Making use of it and supporting incremental computation with intermediate results is an open area for research. Depending on latency prerequisites and throughput of user click stream, one can select streaming or batch processing system.

**Acknowledgements** We would like to thank our project partner Provus Inc. (now part of MasterCard) for providing requirements, insight, and data for this study.

## References

Al-Saleh, K., & Bendak, S. (2013). An Ergonomics Evaluation of Certain ATM Dimensions. *International Journal of Occupational Safety and Ergonomics (JOSE), 19*(3), 347–353.

Amant, R. S., Horton, T. E., & Ritter, F. E. (2007). Model-based evaluation of expert cell phone menu interaction. *ACM Transactions on Computer-Human Interaction (TOCHI), 14*(1), 1:1-1:24.

Apari, T.G., Molu, F., Findik, N., & Dalci, M. (2013). User Experience approach in financial services. *In: Proc. International Conference on Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE)*, pp 400–403

Bailly, G., Oulasvirta, A., Kotzing, T., & Hoppe, S. (2013). MenuOptimizer: Interactive Optimization of Menu Systems. *In: Proc. annual ACM symposium on User interface software and technology (UIST)*, pp 331–341

Cave, K. R., & Wolfe, J. M. (1990). Modeling the role of parallel processing in visual search. *Cognitive Psychology, 22,* 225–271.

Chanco, C., Moquillaza, A., & Paz, F. (2019). Development and validation of usability heuristics for evaluation of interfaces in atms. In A. Marcus & W. Wang (Eds.), *Design, User Experience, and Usability* (pp. 3–18). Berline: Springer.

Cooharojananone, N., Taohai, K., & Phimoltares, S. (2010). A new design of ATM interface for banking services in Thailand. *In: Proc. Annual International Symposium on Applications and the Internet (SAINT)*, pp 312–315

Cremers, A. H. M., de Jong, J. G. M., & Van Balken, J. S. (2008). User-centered design with illiterate persons: The case of the ATM user interface. *Lecture Notes in Computer Science, 5105,* 713–720.

Curran, K., & King, D. (2008). Investigating the human computer interaction problems with automated teller machine navigation menus. *Interactive Technology and Smart Education, 5*(1), 59–79.

Danilenko, A. I., & Goubko, M. V. (2013). Semantic-aware optimization of user interface menus. *Automation and Remote Control, 74*(8), 1399–1411.

Dawe. M. (2007). Understanding mobile phone requirements for young adults with cognitive disabilities. *In: Proc. International ACM SIGACCESS conference on Computers and accessibility*, pp 179–186

Francis, G. (2000). Designing multifunction displays: An optimization approach. *International Journal of Cognitive Ergonomics, 4*(2), 107–124.

Fukazawa, Y., Hara, M., & Ueno, H. (2010). Automatic cell phone menu customization based on user operation history. *Information and Media Technologies, 5*(1), 206–215.

Ghiani, G., Manca, M., Paternò, F., Rett, J., & Vaibhav, A. (2015). Adaptive multimodal web user interfaces for smart work environments. *Journal of Ambient Intelligence and Smart Environments, 7*(6), 701–717.

Goubko, M.V., & Danilenko, A.I. (2010). An Automated Routine for Menu Structure Optimization. *In: Proc. ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS)*, pp 67–76

Hollink, V., van Someren, M., & Wielinga, B. J. (2007). Navigation behavior models for link structure optimization. *User Modeling and User-Adapted Interaction, 17*(4), 339–377.

Huang, H., Yang, M., Yang, C., & Lv, T. (2019). User performance effects with graphical icons and training for elderly novice users: A case study on automatic teller machines. *Applied Ergonomics, 78,* 62–69.

Jain, A. (2012). Optimizing feature-access time through dynamic updates to application menu layout. *ACM SIGSOFT Software Engineering Notes, 37*(5), 1–14.

Karimov, J., & Ozbayoglu, M. (2015). High quality clustering of big data and solving empty-clustering problem with an evolutionary hybrid algorithm. *In: Proc. IEEE International Conference on Big Data*

Karimov, J., Ozbayoglu, M., & Dogdu, E. (2015a). k-means performance improvements with centroid calculation heuristics both for serial and parallel environments. *In: Proc. IEEE International Congress on Big Data*, pp 444–451

Karimov, J., Ozbayoglu, M., Tavli, B., & Dogdu, E. (2015b). Generic menu optimization for multi-profile customer systems. *In: Proc. IEEE International Symposium on Systems Engineering (ISSE)*, pp 163–169

Knuth, D. E. (1985). Dynamic huffman coding. *Journal of algorithms, 6*(2), 163–180.

Kobayashi, H. (1986). Automatic Teller Machine. US Patent No. D283,746

Krishnan, G., Kumar, S., Jithin, C.R., Panicker, V.V., & Sridharan, R. (2011). Service innovation for the user interface of an ATM catering to the needs of the student community. *In: Proc. IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pp 1180–1184

Lee, E., & MacGregor, J. (1985). Minimizing user search time in menu retrieval systems. *Human Factors: The Journal of the Human Factors and Ergonomics Society, 27*(2), 157–162.

Liu, B., Francis, G., & Salvendy, G. (2002). Applying models of visual search to menu design. *International Journal of Human Computer Studies, 56*(3), 307–330.

Matsui, S., & Yamada, S. (2008). A genetic algorithm for optimizing hierarchical menus. *In: Proc. IEEE Congress on Evolutionary Computation,* pp 2851–2858

Mayer, C., Zimmermann, G., Grguric, A., Alexandersson, J., Sili, M., & Strobbe, C. (2016). A comparative study of systems for the design of flexible user interfaces. *Journal of Ambient Intelligence and Smart Environments, 8*(2), 125–148.

McCall, J.A., Richards, P.K., & Walters, G.F. (1977). Factors in Software Quality. Tech. Rep. RADC-TR-77-369, Rome Air Development Center, Air Force System Command, Griffiss Air Force Base, NY

Medhi, I., Patnaik, S., Brunskill, E., Gautama, S. N. N., Thies, W., & Toyama, K. (2011). Designing mobile interfaces for novice and low-literacy users. *ACM Transactions on Computer-Human Interaction, 18*(1), 1–28.

Miller, D.P. (1981). The depth/breadth tradeoff in hierarchical computer menus. *In: Proc. Human Factors and Ergonomics Society Annual Meeting, 25*, pp 296–300

Nielsen, J. (1994). *Usability engineering*. London: Elsevier.

Norman, K.L. (1991). The Psychology of Menu Selection: Designing Cognitive Control at the Human/Computer Interface. Ablex Publishing Corporation

Norman, K. L. (2008). 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.

Norman, K. L., & Chin, J. P. (1988). The effect of tree structure on search in a hierarchical menu selection system. *Behaviour Information Technology, 7*(1), 51–65.

Sears, A., & Shneiderman, B. (1994). Split menus: effectively using selection frequency to organize menus. *ACM Transactions on Computer-Human Interaction (TOCHI), 1*(1), 27–51.

Smyth, B., & Cotter, P. (2003). Intelligent navigation for mobile internet portals. *In: Proc. International Joint Conference on Artificial Intelligence Workshop on Artificial Intelligence,* Information Access, and Mobile Computing, pp 1–8

Taohai, K., Phimoltares, S., & Cooharojananone, N. (2010). Usability comparisons of seven main functions for automated teller machine (ATM) banking service of five banks in thailand. *In: Proc. International Conference on Computational Science and Its Applications (ICCSA),* pp 176–182

Thatcher, A., Shaik, F., & Zimmerman, C. (2005). Attitudes of semi-literate and literate bank account holders to the use of automatic teller machines (ATMs). *International Journal of Industrial Ergonomics, 35*(2), 115–130.

Thimbleby, H. (2000). Analysis and simulation of user interfaces. In: McDonald, S., Waern, Y., Cockton, G. (eds) People and Computers XIV – Usability or Else!, pp 221–237

Troiano, L., & Birtolo, C. (2014). Genetic algorithms supporting generative design of user interfaces: Examples. *Information Sciences, 259,* 433–451.

Troiano, L., Birtolo, C., & Armenise, R. (2016). Searching optimal menu layouts by linear genetic programming. *Journal of Ambient Intelligence and Humanized Computing, 7*(2), 239–256.

Witten, I. H., Cleary, J. G., & Greenberg, S. (1984). On frequency-based menu-splitting algorithms. *International Journal of Man-Machine Studies, 21*(2), 135–148.

Wolsey, L. A. (1998). *Integer Programming*. NewYork: Wiley.

Wolsey, L. A. (2008). Mixed integer programming. In B. Wah (Ed.), *Encyclopedia of Computer Science and Engineering*. NewYork: Wiley.

Zhang, M., Wang, F., Deng, H., & Yin, J. (2012). A survey on human-computer interaction technology for financial terminals. *In: Proc. International Conference on Intelligent Networks and Intelligent Systems (ICINIS),* pp 174–177

## Authors and Affiliations

**Jeyhun Karimov[1] · Murat Ozbayoglu[2] · Bulent Tavli[2] · Erdogan Dogdu[3]**

✉ Erdogan Dogdu
erdogandogdu@gmail.com

Jeyhun Karimov
jeyhun.karimov@dfki.de

Murat Ozbayoglu
mozbayoglu@etu.edu.tr

Bulent Tavli
btavli@etu.edu.tr

1 TU Berlin, Berlin, Germany

2 TOBB University of Economics and Technology, Ankara, Turkey

3 Angelo State University, Texas, USA