An Analytic Model for Time Efficient Personal Hierarchies

William Delamare*
Kochi University of Technology
Kochi, Japan
University of Manitoba
Winnipeg, Canada
William.Delamare@acm.org

Ali Neshati
University of Manitoba
Winnipeg, Canada
neshatia@cs.umanitoba.ca

Pourang Irani
University of Manitoba
Winnipeg, Canada
pourang.irani@cs.umanitoba.ca

Xiangshi Ren
Kochi University of Technology
Kochi, Japan
ren.xiangshi@kochi-tech.ac.jp

ABSTRACT

Hierarchy structures such as file systems are widespread interfaces for item retrieval and selection tasks. Some hierarchies can be modified by end-users, such as application launchers on smartphones or pictures in a file folder. These modifiable hierarchies cannot benefit from an optimization made beforehand as their content, unknown during the design process, is constantly evolving. We hence propose an analytic model which designers can integrate in their system to recommend a range of local structure modifications (e.g., creating new folders) to end-users. Proposing a range of modifications gives flexibility to end-users regarding their own meaningful grouping and labeling choices to follow a recommendation. A first experiment confirms that the recommendations built on our model can lead to modified hierarchies resulting in faster theoretical selection times. A second experiment confirms that the theoretical selection times fit empirical selection times in different hierarchy visual layouts: linear, radial, and grid.

CCS CONCEPTS

• **Human-centered computing → HCI theory, concepts and models**; Graphical user interfaces;

KEYWORDS

Hierarchy; Recommendation; Predictive Model; Modifiable

*JSPS International Research Fellows

CHI 2019, May 4–9, 2019, Glasgow, Scotland UK

ACM ISBN 978-1-4503-5970-2/19/05…$15.00
https://doi.org/10.1145/3290605.3300598

ACM Reference Format:
William Delamare, Ali Neshati, Pourang Irani, and Xiangshi Ren. 2019. An Analytic Model for Time Efficient Personal Hierarchies. In *CHI Conference on Human Factors in Computing Systems Proceedings (CHI 2019), May 4–9, 2019, Glasgow, Scotland UK*. ACM, New York, NY, USA, Article 4, 11 pages. https://doi.org/10.1145/3290605.3300598

1 INTRODUCTION

From audio menus with telephone services to application launchers on wearable devices, file folders and software menus, hierarchical organizations are now widespread to allow for item retrieval and selection. In a hierarchy, the top-level gives access to lower-level groups, which in turn also give access to lower-level groups, and so on until a final option is reached. Hierarchy navigation is still users' preferred information retrieval method, which is consistent, and relies on a recognition task instead of a recall process such as with a search engine [7].

Some hierarchies, such as software menus, have a static content that can be optimized during the design process for fast item selection [6, 23], a critical factor for any retrieval method [13, 15, 19]. Despite the omnipresence of hierarchies with modifiable content, such as digital files and folders, no help is provided to enhance selection times while creating personal hierarchical organizations. This results in end-users having to deal with potentially inefficient hierarchies, or to rely on external features only, such as the 'most frequent items' feature on Microsoft Windows OS for instance.

Let's consider a concrete example. Jane just legally got a new set of digital comical movies (Figure 1). Current optimization approaches mostly consider hierarchies with static content [13, 16, 23]. They can hence assume known constraints, such as item selection frequencies [15, 22, 23] and groupings [12, 13]. This is not the case with modifiable hierarchies and in Jane's situation. Modifying a hierarchy also implies that a familiar structure is already in place [7, 9]. Thus, current approaches involving the complete hierarchy can interfere with any existing mental model and habits of

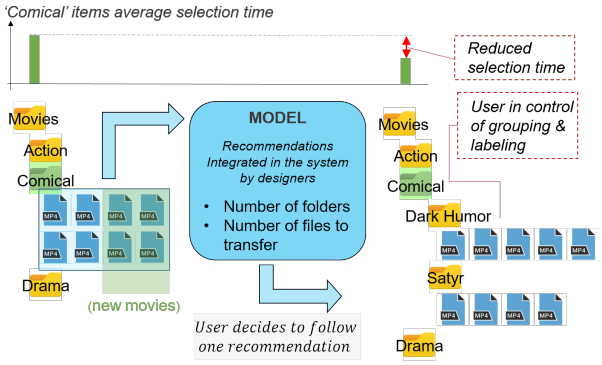

Figure 1: Approach's workflow. The user adds new movies in the 'Comical' folder (left), which increases the average selection time of items in the folder. Our model considers this specific group of items to propose a range of recommendations regarding the number of folders to create and/or the number of files to transfer to reduce this selection time (center). If the user has a grouping and labeling structure in mind that fits a recommendation, the resulting hierarchy (right) will lead to a faster average selection time.

end-users. For Jane, this means that she might want to add these new movies in the existing 'Comical' sub-folder without changing the complete structure of the 'Movie' folder and the other sub-folders. Our model can help end-users optimize their modifiable hierarchies without interfering with their already existing hierarchy structure. For instance, if a designer integrates our model in her interface, it can advise Jane to create between two and four new sub-folders in the 'Comical' folder. She then might be able to find two categories in order to follow this recommendation according to *her* decision (e.g., 'Dark Humor' and 'Satyr', or 'Cringe' and 'Absurd'), giving Jane flexibility regarding grouping and semantic constraints.

Our approach suggests local recommendations to help users organize a part of the hierarchy with newly added items. We focus on modifications following new items addition as it does not interfere with already existing hierarchy parts, and fits with the storing and growing of information content trend [8].

We propose a model to optimize a group of items anywhere in a hierarchy. We then validate two practical aspects of our theoretical approach: its usefulness during hierarchy creation, and its prediction accuracy during item selection. The first experiment confirms that the flexible recommendations built on the model can help users create more time-efficient hierarchies than without recommendations. Participants also expressed their wish to have such recommendations for their personal hierarchies. The second experiment confirms that our theoretical approach fits empirical selection times with different visual hierarchical layouts: linear, radial, and grid. Designers and practitioners offering tools dealing with hierarchies to end-users can now also propose efficient recommendations by integrating our model to their platform. Previous approaches consider the complete hierarchy structure, with static content. Our model can help for *local* structure reorganizations of personal hierarchies with *evolving* content. In addition, our model allows a system to propose not only one (i.e., the optimum), but a *range* of recommendations to give end-users flexibility regarding the new structure to create. End-users are still *in control* of the grouping and labeling of the modified hierarchy. This ensures meaningful choices for future item retrieval.

2 RELATED WORK

We present algorithmic and analytic optimization approaches designed for static menus as none considered modifiable hierarchies.

Algorithmic Approach

Early work proposed a recursive algorithm to determine an optimized hierarchy according to *a priori* defined hierarchies to constrain automatic tree transformations [13]. Matsui et al. proposed a genetic algorithm to optimize hierarchical menus [23]. Their optimization function considers the frequencies of items selection in order to propose item rearrangement. Their model also takes into account the semantic relationships between items and favors balanced hierarchies. Another avenue is to propose design tools based on multi-objective optimization function [6, 22]. The optimization algorithm can then consider several factors such as item position in the menu, groupings, items size, users' expertise, menu learnability, and semantics constraints [22].

These approaches require pre-supposed knowledge, such as example hierarchies [13], or aim to help knowledgeable designers create established software menus [6], allowing control of optimization parameters [27]. These approaches are hence best suited for creating static hierarchies and enhancing menu designs via visual features (e.g., grouping and labeling).

Analytic Approach

Analytic models can help understand the depth/breadth tradeoff of hierarchies highlighted by a broad range of empirical work [10, 18, 19, 25, 28, 33]. The first analytic model concerned hierarchies with homogeneous tree structures, i.e. having a constant breadth, hence the same number of items in each group [19]. The mathematical simulation revealed that groups should contain between 4 and 8 items for ideal selection time. The model has been later revised by considering users experience and menu learnability [28], showing ideal groupings of 16 to 78 items depending on simulation values (e.g. reading time). Further analytic models consider

item selection frequencies [15] or semantic constraints [12]. However, these works are also restricted to uniform trees, i.e. groupings with the same number of children leading to the same frequency distributions.

These models are restricted to specific hierarchy types and concern the whole hierarchical tree structure. In addition, they focus on menus and their broad range of design factors [4]. Finally, and importantly, analytic approaches have not been evaluated to show that the theoretical performance gain fits actual empirical time selection or if end-users would choose these new hierarchies.

Instead, our approach involves local part(s) of modifiable hierarchies, without limitations regarding the hierarchy type (e.g., homogeneous trees). In addition, we empirically validate our approach by demonstrating its usefulness (recommendations) and its generalization (predictions).

3 APPROACH OVERVIEW

For readers not interested in the mathematical aspects of this work, a plain text explanation of our approach is necessary. We consider a node (e.g., a folder) and its children (e.g., movie files) (Figure 2, left). We then simulate the creation of new sub-folders (e.g., 'Satyr') in which some children will be transferred (Figure 2, center). To simplify mathematical manipulations, we consider the number of transferred children constant across sub-folders (e.g., 'Satyr' and 'Absurd') in a given folder (e.g., 'Comical'). We can then compute the difference between the new average selection time and the previous one (without considering errors). This results in an equation depending on 6 parameters: 3 delays (search, select, and transition times), and 3 hierarchy structure values (the original number of children, the number of new sub-folder(s), and their corresponding children). Delays can be obtained via previous work [11]. For instance, pointing selection time can be modeled using Fitts' law [14]. We can then formally analyze *when* the new structure can lead to faster selection times and propose these options to the user (Figure 2, right) - and not only the theoretical optimum. Users can then choose the solution which corresponds to their own grouping and labeling preferences (Figure 2, highlighted row), and proceed to the reorganization of their hierarchy. With this flexibility, Jane has several options to choose from depending on her own grouping and labeling preferences, hence meaningful to her, and potentially helping further browsing [24]. In addition, folders are modeled independently from each other, potentially resulting in specific optimized structures for each of them. For instance, Jane might watch 'Action' movies more often than 'Comical' movies. Thus, the system can consider the expert search time prediction for the 'Action' movies, and the novice one for 'Comical' movies.

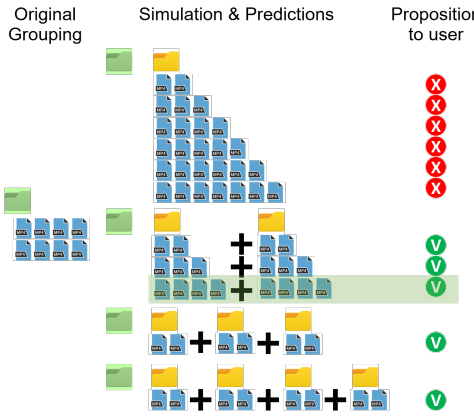

Figure 2: Illustrative example. The model considers a group of nodes (left). The model then simulates and predict the selection time of modified groupings (center). Groupings with a faster selection time can then be proposed to the user (right) for her to make a decision (highlighted row).

4 ANALYTIC APPROACH AND MODEL

This section introduces the main mathematical aspects relevant to recommendations the system can provide. We provide supplementary materials for reader interested in the mathematical details of equations derivations.

Definitions

We consider any node at a level $j-1$ in the hierarchy with n_j children at level j representing items to select, i.e. leaves in the tree. We define three different delays potentially required to browse a hierarchy (Figure 3, left):

- D_1: the delay to search a node while browsing. This can be the time to read file names in a directory for instance.
- D_2: the delay to select a node. This can be the time to move the cursor and click on the desired file for instance.
- D_3: the delay to proceed from the current level to the next one. This can be the steering time (i.e. the time to navigate through a constrained trajectory such as nested-menu levels) between two folders' names when displayed as lists for instance.

Our approach is flexible regarding these delays. For instance, D_1 is related to the time to browse the current hierarchy level. Novice users will perform a *visual search*, with a time linear with the number of items [31]. Expert users will take a *decision*, with a time modeled by the Hick-Hyman law, logarithmic with the number of items [11]. Our model can then be adapted to every specific situations. In addition, these delays could also embed additional information. For instance, if Jane does not like to go through a lot a sub-folders to reach an item (i.e. Jane prefers breadth over depth in the

hierarchy), she could define a cost factor $\alpha_{D_3} > 1$ applied to the transition time, leading to $D'_3 = \alpha_{D_3} \times D_3$. The system would then consider Jane's preferences when proposing to create new sub-folders.

Model

A group of n_j leaves (Figure 3, state 1) will lead to an average selection time of:

$$T_1(j) = T(j-1) + \frac{n_j + 1}{2} D_1 + D_2 \tag{1}$$

If this average selection time of n_j leaves can be reduced, the modified level j will have n'_j nodes, with n'_{j_1} newly created nodes with c_i children each (Figure 3, state 2). The new average selection time is then:

$$T_2(j) = T(j-1) + \frac{1}{n_j} \sum_{i=1}^{n'_j} \Big[max(c_i, 1)(i.D_1 + D_2) + max(c_i, 0)D_3 + \sum_{k=1}^{c_i} (k.D_1 + D_2) \Big] \tag{2}$$

To simplify computations, we consider the number of children c_i constant: $c_i = c, \forall i = 1..n'_{j_1}$. This is obviously not the case in real situations. However, we simplify our computations in *one group of nodes* only. Individual groups can still have a different number of children from each other, thus allowing completely unconstrained tree structures. In addition, with unbalanced node groups, a simple heuristic consists in positioning more frequent groups of nodes before less frequent groups in the hierarchy. If frequencies are unknown - such as with newly added movies - we can assume all items as being equiprobable, and just focus on the number of nodes in each group. Thus, positioning larger groups at the top of the current level reduces the overall selection time. Frequencies can still be represented in our model. For instance, if an item is twice as likely to be selected than others, we can then represent this item with two distinct nodes in its current sub-group, without including any additional mathematical variable. Ordering based on the number of nodes then becomes equivalent to ordering based on item frequencies.

We can now analyze $\Delta_T = T_2(j) - T_1(j)$:

$$\Delta_T = \frac{D_1}{2} \frac{{n'_{j_1}}^2 c}{n_j} (c-1) + \frac{n'_{j_1}}{n_j} \left(c\frac{D_1}{2} + D_2 + D_3 + \frac{D_1}{2} c^2 + n_j D_1 (1-c) \right) \tag{3}$$

First, we explore the theoretical optimum of equation 3. However, the theoretical optimum might not be feasible - nor even be desirable. We hence describe additional recommendations based on different scenarios.

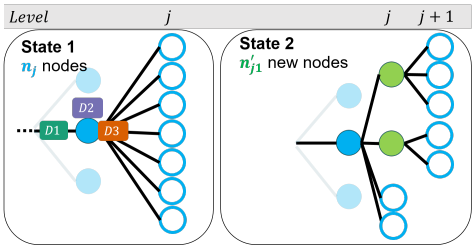

Figure 3: Original (left) and modified (right) hierarchies.

Theoretical Optimum

Jane just added n_j new movies and wants to organize them. The system can suggest what is the best option in this situation.

D_1, D_2, and D_3 are constants defined by the hierarchy layout and interface in use. For a given node to process, the number of children n_j is also constant. Thus, Equation 3 can be seen as a function of only two variables: n'_{j_1} and c, defining a surface. The theoretical optimum is hence obtained when:

$$\frac{\partial \Delta_T}{\partial c} = 0 \quad \text{and} \quad \frac{\partial \Delta_T}{\partial n'_{j_1}} = 0 \tag{4}$$

To propose an optimal solution to the user, equation 4 has to have a solution $(c_0, n'_{j_1 0})$, with $c_0 > 1$, and $n'_{j_1 0} \geq 1$.

Validating an Idea

Jane wants to move N movies into n'_{j_1} new sub-folders. The system can tell if it is worth it.

In this case, we can simply replace c (with $c \approx N/n'_{j_1}$) and n'_{j_1} in equation 3: if the resulting value is negative (i.e. $T_2 < T_1$), the current idea is validated.

In case of a positive value ($T_2 > T_1$), we can go one step further by providing insight to the user about why the idea is not worth it given the current number of nodes considered in the group. This extra-information is possible if we consider the current number of nodes n_j flexible [6, 28]. Indeed, some nodes could be categorized in more than one group (e.g., a movie can be in 'Action' and 'Comical'). If one or more extra nodes could have been part of the current group, the user knows that she can revise her current categorization. With c and n'_{j_1} set as constant in equation 3, we can now determine what would be the minimum number of nodes n_j that the level should have for the idea to be validated:

$$n_j > \frac{c}{1-c} \left(\frac{n'_{j_1}}{2}(1-c) - \frac{c}{2} - \frac{\frac{D_1}{2} + D_2 + D_3}{D1} \right) \tag{5}$$

Finding the Number of Children

Jane wants to create n'_{j_1} new sub-folders to organize some of her newly added movies. The system can suggest a range regarding how many movies should be in each one.

By fixing n'_{j_1} in Equation 4, we can find the theoretical best option for c as we did for the theoretical optimum, defined

by:

$$c_0 = \frac{n_j}{n'_{j_1} + 1}\left(\frac{n'_{j_1}}{2n_j} + 1 - \frac{\frac{D_1}{2} + D_2 + D_3}{D_1 n_j}\right) \quad (6)$$

We can go one step further and propose a range of values for the number of children the newly created nodes should have to make the new hierarchy more time efficient. By re-writing equation 3 as a second degree polynomial function, we can then compute the roots c_1 and c_2 of the polynomial function to find a range of values for which $\Delta_T < 0$.

Finding the Number of Groups

Jane wants to move N movies from the current folder. The system can suggest a range regarding how many new sub-folders should be created.

Let $k \in [0; 1]$, so that $N = n'_{j_1} c = k n_j$. In this case, the optimum number of groups is obtained for:

$$n'_{j_1} = k\sqrt{\frac{n_j}{2-k}} \quad \text{giving:} \quad c = \sqrt{(2-k)n_j} \quad (7)$$

It has been previously shown that the optimal number of groups to divide n_j nodes is to create $n'_{j_1} = \sqrt{n_j}$ new nodes [28]. This is true only if we consider that users want to send all n_j nodes to the next level, i.e. when $k = 1$. Our analysis provides more flexibility to users by considering only a sub-sample of nodes, i.e. $0 < k \leq 1$. In other words, in a given folder, sub-folders can co-exist with files.

We can go one step further if we consider the number of nodes n_j flexible again [6, 28]. We can then find n_{j_n} for which $\Delta_T < 0$:

$$n_{j_n} = \left(\frac{D_1 + \sqrt{{D_1}^2 + 2D_1(\frac{D_1}{2} + D_2 + D_3)}}{D_1\sqrt{2-k}}\right)^2 \quad (8)$$

This is again motivated by potentially flexible semantic constraints. Thus, we can give further suggestions regarding the number of nodes in the current group to ensure the proposed modification will lead to time efficient selections.

5 VALIDATION 1: RECOMMENDATIONS

We want to know if our approach can be applied with restrictions such as semantic constraints and personal preferences: can users follow recommendations, and if so, to which extent?

We provided two interfaces: with and without help. With both interfaces, participants had to create hierarchies represented in a tree-like structure with node displayed as circles and labels, and the hierarchy by links between nodes. Participants were able to create and to name new nodes, to delete created nodes (not the nodes from the data sets), to move node, and to transfer nodes as children using drag-and-drop mouse interactions. A red link indicated that either (i) a group node had no children, or (ii) a node containing a word from the data set had a child. Participants could not validate a design with red links. Both interfaces displayed a legend, the average selection time of the current hierarchy, a button to sort the tree (large groups positioned first), a button to reset the tree, and links directing users to the definition of the words used in the experiment. The only difference between with and without help concerned recommendations. With help, groups of nodes that could be optimized had green links, and recommendations involved:

- Theoretical optimum (label based on equation 4). The system displays how many new nodes should be created and how many existing nodes should be transferred on each of them to obtain the ideal hierarchy.
- Idea validation (form based on equations 3 and 5). Users can input the number of node(s) they want to create and the number of children. If the idea is not validated (i.e. longer selection time than with the current hierarchy), the system displays the minimum number of nodes the group should have for the idea to be worth it.
- Number of children (form based on equation 6). Users can input the number of groups they think about creating. The system displays the optimal number of children each new group should have and the range of valid solutions.
- Number of groups (form based on equations 7 and 8). Users can input the number of nodes they consider sending to a new level. The system displays the optimal number of group to create or the number of nodes the current level should have for this idea to be worth it.
- Overall performance (table based on equation 2). A color-coded table showed users all combination of number of groups/children and expected gain of time. The table was sync to results from input forms to ease the navigation.

Task

We asked participants to create hierarchies with and without help. We proposed two data sets of words by considering 170 animals, 100 house items, 70 movies, and 126 famous people, which explicitly enforced unbalanced hierarchies. We then randomly generated 2 data sets of 60 unique words each and random delays. Recommendations would help participants to recursively optimize newly created group of nodes until no group could be optimized anymore, or until participants decided to not follow the remaining recommendations.

Participants and Apparatus

We recruited 16 participants (8 females), aged 18 to 56 ($M = 26.7, SD = 9.3$). We gave participants $15 gift cards. All had experience with hierarchical organization on desktop computer - the example we used to describe the task. The

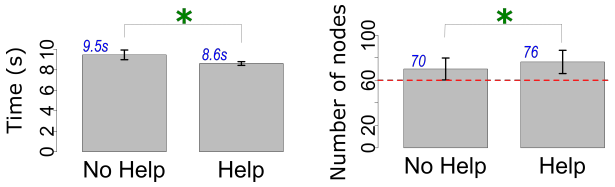

Figure 4: Left: Average item selection time (s). Right: Total number of nodes. Error bars show 95% CI (N=16).

online tool ran on a 3.6 GHz Intel Core i7 computer via Chrome Web Browser on a 24″ desktop screen.

Procedure

The experiment lasted between 45min and 1h30 depending on the participant's knowledge of the words used in the data sets, after which participants filled a questionnaire for qualitative results. The session started with an introduction to hierarchies (e.g., menu and file folders) and both tools. Without recommendation, we instructed participants to create hierarchies as they would do on their own. With recommendations, we instructed participants to *try* to follow the recommendations while considering meaningful grouping.

Experimental Design

The task used a within-participants design with *System* (No Help, Help) as an independent variable and counterbalanced across participants. Each participant used one version of the system with only one dataset once. To fully control the learning effect, we also counterbalanced the presentation of the pairs *System/Dataset*. The experiment was divided into 2 blocks: one for each pair *System/Dataset*. Each block started with a 8-item data set practice trial. The 60 words were sorted alphabetically in order to avoid any explicit grouping.

Results

Our data did not satisfy both the normality and the homogeneity of variances assumptions. We performed our analysis with Wilcoxon non-parametric tests. We made sure there was no significant effect of *Dataset* [$W = 58$, $Z = -0.52$, $p = 0.63$] on the average selection time. We hence aggregated the data by *System* and participants to complete our statistical analysis.

Average Selection Time. We found a significant effect of *System* on the average selection time with a large effect size [$W = 133$, $Z = 3.36$, $p < 0.001$, $r = 0.59$]: Without recommendation, the resulting average selection time is 9.9% slower than with recommendations (Figure 4, left). This indicates that participants could benefit from recommendations even with practical constraints.

Hierarchy Structure. Participants created significantly more nodes with recommendations than without [$W = 17$, $Z = -2.45$, $p < 0.05$, $r = 0.45$] (Figure 4, right). This was to be expected, as we built the theoretical model to help *create* new nodes. Note that participants also chose to sometimes not to follow recommendations. The average gain of time not followed by participants was 0.89s without help (unknown), and 0.03s with help (known).

12 out 16 participants created hierarchies in which leaves cohabited with sub-groups of nodes at various depth levels. This validates that providing a flexible way to arrange nodes ($k \neq 1$ in Equation 7) is actually useful in scenarios with semantic constraints.

Qualitative Results. 11/16 participants would like such a recommendation system to organize their files. 12 participants still realized the recommendation system could be "useful" and "helpful" (P4, P5, P6, P13), and the experimental tool "easy to use" (P3, P14), even if some participants felt the recommendation system was "slowing them down" (P5) or interfering with their own categorization scheme (P4, P6, P14).

6 VALIDATION 2: PREDICTION

We showed that our approach can successfully reduce the theoretical selection time. We next want to investigate if our approach is conform to real empirical selection times. More specifically, we are interested in (1) validating the selection time prediction of the model with different common item layouts (i.e. linear, radial, and grid), and (2) investigating the feasibility of our heuristic to overcome unbalanced nodes groups. We hence collect pointing and gaze data while users perform item selections.

Item Layouts

We consider three common item layouts: *linear*, *radial*, and *grid* (Figure 5). Linear layouts are widespread in software applications (e.g., menus) [5, 6, 21]. Radial layouts are also a well-known layout (e.g., pie-menus) [29, 33]. Grid layouts are also common [3], especially on mobile devices [9].

We focus on 2-levels hierarchies, as our approach considers only a level and its children (equation 3). The sub-level (i) appears on the right of the selected item with the linear layout, the top of the sub-level aligned with the previously selected item, (ii) appears centered around the cursor position with the radial layout, and (iii) replaces the previous level with the 2 by 2 grid layout. Users can navigate between panels by clicking on arrows at the top and bottom of the screen.

Calibration: Delay Parameters

Recall that our model requires the delays associated with visual search (D_1), selection (D_2), and transition between levels (D_3). Fortunately, previous works provide several models

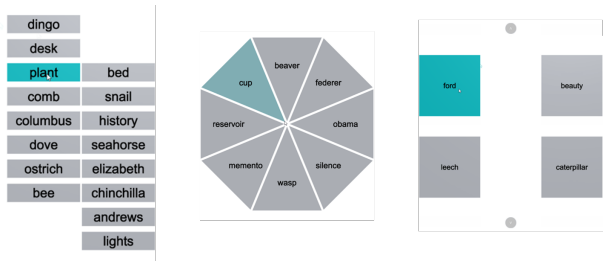

Figure 5: Linear (2 levels of 8 items each), radial, and grid layouts. Blue elements indicate mouse cursor hover states.

to do so [5, 11, 18, 20]. We chose to determine D_1, D_2, and D_3 using the *Search, Decision, and Pointing* model (SDP) [11], which has been proven effective to model selection times in different menu types [3, 10]. The SDP model is built on other low-level models, such as Fitts' Law to predict the pointing time [14], the Hick-Hyman law to predict the decision time [17], and the steering law to predict the navigation time between menu levels [1]. Once D_1, D_2, and D_3 are calibrated via the SDP model, we can assess how well equation 2, the core basis of our approach, fits empirical data based on structural parameters: the original number of children, the number of new sub-folder(s), and their corresponding children.

D_1: Delay to Process a Node. D_1 is related to the time to browse the current hierarchy level while looking for the desired item. For novice users, this browsing time corresponds to a *visual search* time, linear with the number of items [31]. For expert users, this browsing time corresponds to a *decision* time modeled by the Hick-Hyman law, logarithmic with the number of items [11]. In our experiment, participants will be novices. We hence use the novice linear relationship between the visual search time T_{vis} and the number of items N_k in the k^{th} level: $T_{\text{vis}} = a_{\text{vis}} + b_{\text{vis}} \times N_k$, with a_{vis} and b_{vis} empirically determined constants. Since D_1 is the time to process *one* item, we define $D_1 = \frac{T_{\text{vis}}}{N_k}$. However, we want D_1 to be specific to a layout type, not a hierarchy level k. We hence chose to approximate D_1 - the delay to process *any* node in the hierarchy - by considering the average number of nodes N_{avg}:

$$D_1 = \frac{a_{\text{vis}}}{N_{\text{avg}}} + b_{\text{vis}} \tag{9}$$

Note that this approximation might lead to over- and under-estimation in case a level has less / more nodes than N_{avg}.

D_2: Delay to Select a Node. In our case, users select an item by pointing and clicking at it. The pointing task is commonly modeled using Fitts' Law, stating that the time T_p required to point at a target is linear with the Index of Difficulty ID: $T_p = a_p + b_p \times ID$, with a_p and b_p empirically determined constants. For most layouts, ID depends upon the target width W and the movement amplitude A: $ID = \log(\frac{A}{W} + 1)$. However, for radial layouts, the pointing time follows a linear function of the number of items N_k in the k^{th} level [3]: $ID = N_k$. T_p represents the average pointing selection time, but depends upon W and A, or k. We want D_2 to be independent of any other factors than the empirically determined factors a_p and b_p to characterize the layout. We hence approximate D_2 - the delay to select *any* node - by:

$$D_2 = a_p + b_p \times \text{mean}(ID) \tag{10}$$

As for D_1, this can also lead to over- and under-estimation.

D_3: Delay to Transition Between Hierarchy Levels. Navigating through multiple levels in linear layouts is often modeled via the steering law, which predicts the transition time T_{tr} of a movement of amplitude A_t in a tunnel of width W_t as $T_{\text{tr}} = a_{\text{tr}} + b_{\text{tr}} \times \frac{A_t}{W_t}$, with a_{tr} and b_{tr} empirically determined constants. For radial layouts, we consider the transition time to be $T_{\text{tr}} = 0$, as the next level appears centered on the cursor position, removing the need for any steering movement. This illustrates how different layout properties are reflected in our model. For our grid layout, the sub-level replaces the current level. We hypothesize that the average transition cursor movement can be approximated by an average pointing time $T_{\text{tr}} = a_{\text{tr}} + b_{\text{tr}} \times \text{mean}(ID)$. For all layouts, we approximate the steering task with the pointing parameters (i.e. $a_{\text{tr}} = a_p$, $b_{\text{tr}} = b_p$), and then consider:

$$D_3 = T_{\text{tr}} \tag{11}$$

Participants and Apparatus

We recruited 21 participants (10 females), aged 18 to 56 (M=25.3, SD=7.8). We gave participants $15 gift cards. We carried out our experiment on a 3.6GHz Intel Core i7 desktop computer, with a 24" (1920 × 1080) screen. We used an Eye Tribe eye tracker, with a 60Hz sampling rate and a gaze estimation error between 0.5° and 1° according to the manufacturer. The software was implemented in C# with the Unity3D 5.3 game engine.

Procedure

The experiment lasted around 1h per participant. The session started with a 9-points calibration for the eye tracker. Each participant was assigned to one layout (linear, radial, or grid).

The session was divided into two blocks: one for a Fitts task, and one for an item selection task. The Fitts task ensured we captured data to derive the pointing selection delay D_2 without visual search. Participants started a Fitts trial by clicking a grey 'start' button to make the green target appear. Participants had to select the green target as fast and as accurately as possible. The trial stopped when the target was clicked. The item selection task consisted in two consecutive selections in a specific layout. Participants could read the sequence of items to find and select in the menu (e.g., "plant > chinchilla", Figure 5-left) as long as they wanted.

They could then press the start button to display the first hierarchy level. If participants clicked the correct first item (e.g., 'plant'), the second level appeared. If participants selected an incorrect item, the trial stopped and was counted as an error. If the eye tracker lost the gaze for more than 2.5s, the cursor disappeared and participants had to re-position themselves.

Layout Parameters

The linear layout used 230px × 35px labels. For both Fitts and the item selection tasks, the start button was positioned at the top-left of the invisible first level. For the Fitts task, participants selected the first, middle and last items of both levels, creating 6 IDs: 2.93, 3.46, 3.83, 4.02, 4.21, and 4.45.

The radial layout used labels centered in 220px radius slices. For both Fitts and the item selection tasks, the start button was positioned at the center of the circular layout. For the Fitts task, we were primarily interested in the number of slices instead of the actual Fitts ID [3]. Thus, participants had to select 6 targets, i.e. a random single slice in a layout corresponding to a discretization of 4, 6, 8, 10, 12, and 16 slices.

The grid layout used 275px × 275px labels and 110px × 110px navigation arrows buttons. For the Fitts task, we considered the 6 combinations of movement between the four layout elements, an extra center button (184px × 60px), and the bottom navigation arrow, leading to the IDs 1.14, 1.16, 1.59, 1.72, 1.77, and 2.17. For the item selection task, the start button was in the center of the screen.

For all layouts, we considered first levels of 8, 12, and 16 items. We randomly generated two second levels for each first levels, leading to 6 different instances of each layout. Each first level node could have 4, 8, or 12 children. We chose multiple of 4 to avoid any side-effect due to potential gap in the grid layout. We used labels from the first experiment categories (animals, etc), but only considered single words, which were ±2 letters from the average word length to avoid any side-effect due to potential salience.

Experimental Design

Data from each layout (Linear, Radial, Grid) were analyzed separately. Fitts' Law parameters (for D_2 and D_3) were calibrated using Fitts' block data. The dependent pointing time variable T_p corresponds to the time between the start button click and the green target click. Visual search time parameters (for D_1) were calibrated using the item selection task block data. The dependent visual search time variable T_{vis} corresponds to the time between the level appearance and the time the gaze reached the item (averaged across all gaze hits on the item during the trial). We then used the determined D_1, D_2, and D_3 to estimate how well our approach can model item selection times via regression analyses of Equation 2 across layout instances.

Table 1: Regression analysis results for both pointing (Fitts) and visual search (Search) calibrations, the resulting delays, and regression analysis results for the model absolute prediction (Pred.) and the visual search ordering of items (Order).

Layout		a	b	adj. R^2	F-Stat.	p
Linear	Search	0.33	0.11	.99	2782	$< .001$
	Fitts	0.25	0.14	.61	8.88	$< .05$
	Delays	$D_1 = 0.12$, $D_2 = 0.76$, $D_3 = 0.95$				
	Pred.	-3.33	1.9	.91	53.07	$< .01$
	Order	58.25	11.32	.98	879.9	$< .001$
Radial	Search	0.45	0.09	.98	174.2	$< .01$
	Fitts	0.44	0.01	.86	32.21	$< .001$
	Delays	$D_1 = 0.11$, $D_2 = 0.53$, $D_3 = 0.0$				
	Pred.	-0.22	1.93	.93	68.62	$< .001$
	Order	105.72	0.35	-.05	0.29	0.60
Grid	Search	0.0	0.23	.99	13690	$< .001$
	Fitts	0.24	0.2	.77	17.49	$< .05$
	Delays	$D_1 = 0.25$, $D_2 = 0.53$, $D_3 = 0.53$				
	Pred.	-1.69.	1.77	.98	231.3	$< .001$
	Order	31.95	22.5	.96	385.8	$< .001$

During the Fitts block, participants had to perform 15 selections of each target, leading to 6 targets × 15 selections × 21 participants = 1890 acquisitions. During the item selection block, participants had to perform 50 correct selections in each layout instance, leading to 50 selections × 6 instances × 21 participants = 6300 acquisitions.

Results

We first discuss the visual search and pointing times calibrations, and the resulting delays associated with each layout. We then focus on the accuracy of the model prediction, and the visual search ordering. We removed 694 trials for which the system lost the gaze and froze the mouse cursor. Numerical data can be found in Table 1.

Calibration. **Visual search** results hold a strong linear correlation between the visual search time and the number of items (all adjusted $R^2 > .98$), which is in line with previous work [3, 11, 30]. **Pointing** results are not as strong as in previous work for the Linear layout [11] (adjusted $R^2 = .61$), compared to the grid layout (adjusted $R^2 = .77$). However, results confirm the linear relationship between the pointing selection time and the number of slices in the radial layout (adjusted $R^2 = .86$) [3]. Results for the linear layout might be explained by the fact that we considered only the targets height [2], or the fact that we did not apply any adjustment for accuracy [32] to name a few. However, the delays of our model are based on approximations of human-motor models,

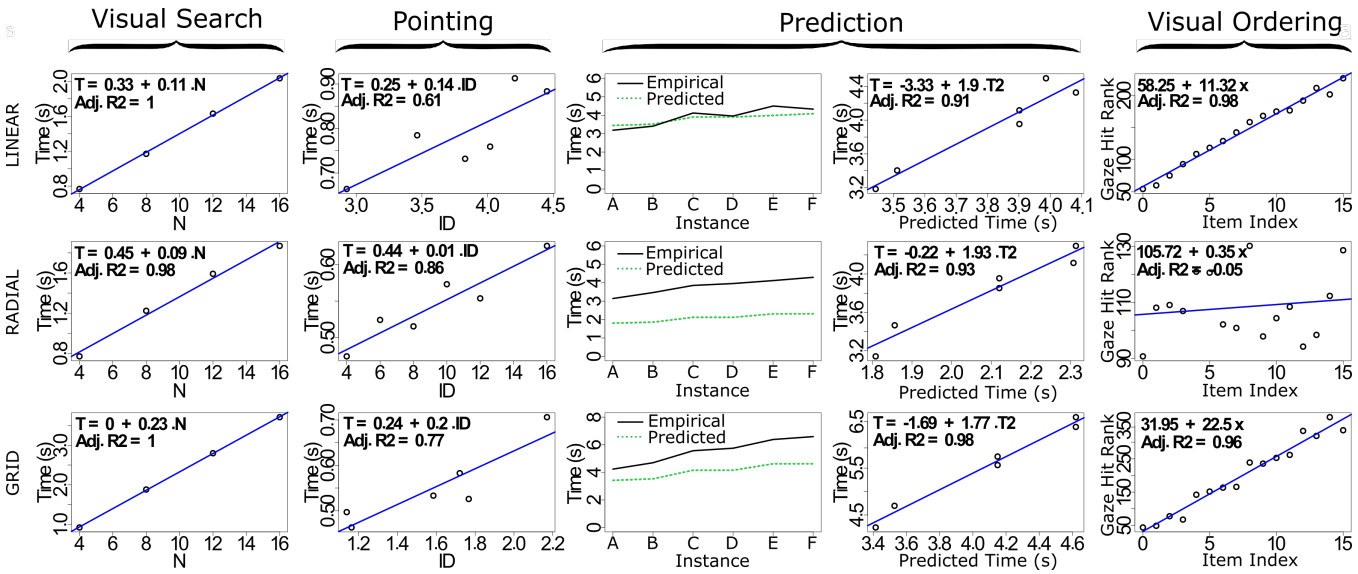

Figure 6: Top to bottom: Linear, Radial, Grid. Left to right: Visual search, pointing, absolute prediction, and visual search ordering.

themselves approximations of the reality. We hence want to validate our model even with rough empirical parameters estimates.

Delays reveal differences between layouts. D_1 shows that the Grid layout imposes a longer processing time than others. This can be explained by the navigation through different panels limited to 4 items per panel. D_2 shows that the linear layout imposes a slower selection time than the others. This results is most likely due to the fact that the linear layout uses smaller labels compared to the others, combined to the fact that other layouts appear centered around the cursor. Note that although we found a correlation between eye-movement and cursor movements - confirming that the cursor moves as participants look for a label [5] - this selection time remains slower for the Linear layout. D_3 shows that the transition time between hierarchy levels is higher for Linear than Grid, most likely because of the tunneling action of the cursor.

Prediction. Our model successfully predicts performance trends across hierarchy instances, with an adjusted R^2 of 0.95 for Linear, 0.82 for Radial, and 0.98 for Grid (Table 1, Figure 6). This validates the fact that Equation 2 can be used to predict the selection time of items in a two-levels hierarchy. Thus, the root of other equations (such as Equation 3) is empirically validated. Prediction made from these equations are then in line with empirical data.

However, the *absolute* prediction is not perfect: Linear regressions show that $T_{\text{real}} = a + b \times T_{\text{pred}}$, with $a \neq 0$ and $b \neq 1$. We discuss this underestimation of empirical data in the next section, but we note that our model is used to compare hierarchies *relatively* to each other, not in an *absolute* way.

Visual Search Ordering. Our model uses a simplification by considering the number of children constant. A simple heuristic - assuming equiprobable item selection - consists in positioning larger groups of nodes 'at the top' of the hierarchy level. We hence want to verify if such 'top' concept exists for our hierarchy layouts.

We found a strong linear correlation between item position and gaze path for the Linear (adjusted $R^2 = .98$) and Grid (adjusted $R^2 = .96$) layouts, with the gaze going from top-to-bottom, left-to-right. However, we did not find any correlation for the Radial layout (adjusted $R^2 = -.05$) when considering a clockwise order of slices, 0 starting at 12h. Previous work found a correlation with a succession of left/right alternations. However, we could not replicate this result (adjusted $R^2 = -.06$). We found a pattern which consisted in a broad exploration of the left side, followed by sequences of 2 items on the right / 1 item on the left, but this result remains to be further explored.

7 LIMITATIONS AND FUTURE WORK

We showed that our model could effectively lead to time-efficient hierarchies (experiment 1). We further validated our approach by showing that predicted trends in theoretical selection times between hierarchies had a strong correlation to actual empirical data with different hierarchy layouts (linear, radial, and grid) (experiment 2). We now discuss findings and implications of the present work.

Recommendations: We provide evidence that recommendations can be useful, but several interesting questions

still remain. For instance, we did not evaluate how recommendations could be presented or how would users adapt their behavior to such system. *In-vivo* longitudinal studies could provide answers regarding the actual impact of recommendations and their presentation.

Predictions: We showed that our model could provide a good prediction of selection time performances in hierarchies *relatively* to each other. This was the goal of our approach in order to provide recommendations to end-users when modifying these hierarchies. However, the *absolute* prediction is off by approximately 1s, like in previous work [3], which hypothesized that this *absolute* underestimation could be attributed to their experimental protocol and/or their use of a timeout trigger method. The fact that our work also shows an offset of ∼1s makes us think that the problem might be reproducible, and hence investigated and explained.

Frequency and visual ordering: We considered sub-groups with a constant number of children for a given group of nodes. These sub-groups have different selection frequency. In our experiments with equiprobable items, this frequency was determined by the number of nodes in the sub-group. If frequencies are available, the model can include this aspect by duplicating each node according to their frequency. We hypothesized that more frequent sub-groups should be positioned 'first' in the hierarchy. However, we could determine such ordering concept only for the linear and grid layouts, not for the radial one. Although this does not prevent accurate relative predictions, users still need to position frequently accessed sub-groups 'first' by themselves (e.g., to not scroll every time to access a frequent sub-folder).

Delays (reverse approach): We showed that calibrating our model and the delays via SDP could induce comparisons between hierarchy layouts. We could reverse our approach. In this work, we empirically determine delays to analyze our equation based on our three parameters of interests regarding the hierarchy structure (namely the original number of children, the number of new sub-folder(s), and their corresponding children). Instead, we could fix the hierarchy structure, and analyze our model according to the delays. For instance, the system could determine which delays (and hence which corresponding layout) better fits a given hierarchy structure.

Errors: There is still no clear model regarding *errors* probabilities during a selection task in a hierarchy [15, 26]. In addition, some works consider the probability of performing selection errors as level-independent [19], i.e. the same error probabilities in every levels of the hierarchy, while some other works consider it level-dependent [28]. If the later is true, then integrating extra cost time due to errors in our level-agnostic model will be difficult. We hence envision a modular approach: a module based on our model for time selections, and a module based on an error model - which remains to be derived and experimentally validated. Both modules could then be combined to also anticipate delays due to memory errors after a modification of a hierarchy. These modules could both be added as a complementary part to previous work approaches, which focus on before-hand optimization of the complete hierarchy.

8 CONCLUSION

We propose a model to help end-users optimize the selection time when adding new content such as new files in a directory. Despite the predominance of modifiable hierarchies in our everyday lives, no work considered a localized approach that could help end-users to optimize the newly added content only, without interfering with their existing hierarchy structure.

Our model allows to recommend *if*, *when*, and *how* users can optimize any hierarchy in different scenarios. We validate our approach with two user experiments. The first one confirms that our flexible recommendations can effectively be used to design time-efficient hierarchies. In addition, such recommendation system is well-accepted by end-users. The second experiment confirms that our model can effectively predict the difference in selection time between hierarchies. The advantage of our approach compared to previous analytic models is that we do not constrain the overall tree structure to a specific type. The advantage of our approach compared to previous algorithmic approaches is that users are still in charge of the grouping and labeling process. This can enhance memorability for future browsing of the newly modified structure. Our model can be a complementary solution to previous approaches that consider the whole hierarchical structure at first. Designers can embed our model in their system to help end-users when the original hierarchy content is altered.

9 ACKNOWLEDGMENTS

We acknowledge the support from the NSERC CRC program, the NSERC Discovery grant, as well as the JSPS fellowship program. We also thank Antti Oulasvirta for his valuable comments, and Wang Chen for the video.

REFERENCES

[1] Johnny Accot and Shumin Zhai. 1997. Beyond Fitts' law: models for trajectory-based HCI tasks. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '97*. ACM Press, New York, New York, USA, 295–302. https://doi.org/10.1145/258549.258760

[2] Johnny Accot and Shumin Zhai. 2003. Refining Fitts' law models for bivariate pointing. In *Proceedings of the conference on Human factors in computing systems - CHI '03*. ACM Press, New York, New York, USA, 193–200. https://doi.org/10.1145/642611.642646

[3] David Ahlström, Andy Cockburn, Carl Gutwin, and Pourang Irani. 2010. Why it's quick to be square: Modelling new and existing hierarchical menu designs. In *Proceedings of the 28th international conference*

on Human factors in computing systems - CHI '10. ACM Press, New York, New York, USA, 1371–1380. https://doi.org/10.1145/1753326.1753534

[4] Gilles Bailly and Antti Oulasvirta. 2014. Toward Optimal Menu Design. *ACM Interactions* (2014), 40–45. https://doi.org/10.1145/2617814

[5] Gilles Bailly, Antti Oulasvirta, Duncan P Brumby, and Andrew Howes. 2014. Model of visual search and selection time in linear menus. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI '14*. ACM Press, New York, New York, USA, 3865–3874. https://doi.org/10.1145/2556288.2557093

[6] Gilles Bailly, Antti Oulasvirta, Timo Kötzing, and Sabrina Hoppe. 2013. MenuOptimizer: Interactive Optimization of Menu Systems. In *Proceedings of the 26th annual ACM symposium on User interface software and technology - UIST '13*. ACM Press, New York, New York, USA, 331–342. https://doi.org/10.1145/2501988.2502024

[7] Ofer Bergman, Ruth Beyth-Marom, Rafi Nachmias, Noa Gradovitch, and Steve Whittaker. 2008. Improved search engines and navigation preference in personal information management. *ACM Transactions on Information Systems* 26, 4 (sep 2008), 1–24. https://doi.org/10.1145/1402256.1402259

[8] Richard Boardman and M Angela Sasse. 2004. "Stuff goes into the computer and doesn't come out" A Cross-tool Study of Personal Information Management. In *Proceedings of the 2004 conference on Human factors in computing systems - CHI '04*, Vol. 6. ACM Press, New York, New York, USA, 583–590. https://doi.org/10.1145/985692.985766

[9] Matthias Böhmer and Antonio Krüger. 2013. A study on icon arrangement by smartphone users. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '13*. ACM Press, New York, New York, USA, 2137–2146. https://doi.org/10.1145/2470654.2481294

[10] Andy Cockburn and Carl Gutwin. 2009. A Predictive Model of Human Performance With Scrolling and Hierarchical Lists. *Human-Computer Interaction* 24, 3 (jul 2009), 273–314. https://doi.org/10.1080/07370020902990402

[11] Andy Cockburn, Carl Gutwin, and Saul Greenberg. 2007. A predictive model of menu performance. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '07*. ACM Press, New York, New York, USA, 627–636. https://doi.org/10.1145/1240624.1240723

[12] A I Danilenko and M V Goubko. 2013. Semantic-aware optimization of user interface menus. *Automation and Remote Control* 74, 8 (aug 2013), 1399–1411. https://doi.org/10.1134/S000511791308016X

[13] Donald L Fisher, Erika J Yungkurth, and Stanley M Moss. 1990. Optimal Menu Hierarchy Design: Syntax and Semantics. *Human Factors: The Journal of the Human Factors and Ergonomics Society* 32, 6 (dec 1990), 665–683. https://doi.org/10.1177/001872089003200605

[14] P M Fitts. 1954. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of experimental psychology* 47 (sep 1954), 381 – 391. https://doi.org/10.1037/h0055392

[15] Mikhail V Goubko and Alexander I Danilenko. 2010. An automated routine for menu structure optimization. In *Proceedings of the 2nd ACM SIGCHI symposium on Engineering interactive computing systems - EICS '10*. ACM Press, New York, New York, USA, 67–76. https://doi.org/10.1145/1822018.1822030

[16] M V Goubko and A I Danilenko. 2012. Mathematical model of hierarchical menu structure optimization. *Automation and Remote Control* 73, 8 (aug 2012), 1410–1423. https://doi.org/10.1134/S0005117912080140

[17] Ray Hyman. 1953. Stimulus information as a determinant of reaction time. *Journal of Experimental Psychology* 45, 3 (1953), 188–196. https://doi.org/10.1037/h0056940

[18] T K Landauer and D W Nachbar. 1985. Selection from alphabetic and numeric menu trees using a touch screen. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '85*. ACM Press, New York, New York, USA, 73–78. https://doi.org/10.1145/317456.317470

[19] Eric Lee and James Macgregor. 1985. Minimizing User Search Time in Menu Retrieval Systems. *Human Factors: The Journal of the Human Factors and Ergonomics Society* 27, 2 (apr 1985), 157–162. https://doi.org/10.1177/001872088502700203

[20] Baili Liu, Gregory Francis, and Gavriel Salvendy. 2002. Applying models of visual search to menu design. *International Journal of Human-Computer Studies* 56, 3 (mar 2002), 307–330. https://doi.org/10.1006/ijhc.2002.0527

[21] Wanyu Liu, Gilles Bailly, and Andrew Howes. 2017. Effects of Frequency Distribution on Linear Menu Performance. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI '17*. ACM Press, New York, New York, USA, 1307–1312. https://doi.org/10.1145/3025453.3025707

[22] Zhengwu Lu. 2017. Optimization Approaches to Adaptive Menus. May (2017).

[23] Shouichi Matsui and Seiji Yamada. 2008. Genetic algorithm can optimize hierarchical menus. In *Proceeding of the twenty-sixth annual CHI conference on Human factors in computing systems - CHI '08*, Vol. 1. ACM Press, New York, New York, USA, 1385–1388. https://doi.org/10.1145/1357054.1357271

[24] Craig S Miller and Roger W Remington. 2004. Modeling Information Navigation: Implications for Information Architecture. *HumanâĂŞComputer Interaction* 19, 3 (sep 2004), 225–271. https://doi.org/10.1207/s15327051hci1903_2

[25] Dwight P. Miller. 1981. The Depth/Breadth Tradeoff in Hierarchical Computer Menus. *Proceedings of the Human Factors Society Annual Meeting* 25, 1 (oct 1981), 296–300. https://doi.org/10.1177/107118138102500179

[26] Kent Norman. 1992. The Psychology of Menu Selection: Designing Cognitive Control of the Human/Computer Interface. *Displays* 13 (1992).

[27] Antti Oulasvirta. 2017. User Interface Design with Combinatorial Optimization. *Computer* 50, 1 (jan 2017), 40–47. https://doi.org/10.1109/MC.2017.6

[28] Kenneth R Paap and Renate J. Roske-Hofstrand. 1986. The Optimal Number of Menu Options per Panel. *Human Factors: The Journal of the Human Factors and Ergonomics Society* 28, 4 (aug 1986), 377–385. https://doi.org/10.1177/001872088602800401

[29] Krystian Samp and Stefan Decker. 2010. Supporting menu design with radial layouts. In *Proceedings of the International Conference on Advanced Visual Interfaces - AVI '10*. ACM Press, New York, New York, USA, 155–162. https://doi.org/10.1145/1842993.1843021

[30] Krystian Samp and Stefan Decker. 2011. Visual search in radial menus. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 6949 LNCS. 248–255. https://doi.org/10.1007/978-3-642-23768-3_21

[31] Andrew Sears and Ben Shneiderman. 1994. Split menus: effectively using selection frequency to organize menus. *ACM Transactions on Computer-Human Interaction* 1, 1 (mar 1994), 27–51. https://doi.org/10.1145/174630.174632

[32] R. William Soukoreff and I. Scott MacKenzie. 2004. Towards a standard for pointing device evaluation, perspectives on 27 years of Fitts' law research in HCI. *International Journal of Human-Computer Studies* 61, 6 (dec 2004), 751–789. https://doi.org/10.1016/j.ijhcs.2004.09.001

[33] Shengdong Zhao, Maneesh Agrawala, and Ken Hinckley. 2006. Zone and polygon menus: using relative position to increase the breadth of multi-stroke marking menus. In *Proceedings of the SIGCHI conference on Human Factors in computing systems - CHI '06*. ACM Press, New York, New York, USA, 1077–1086. https://doi.org/10.1145/1124772.1124933