Comparative Analysis of Deep vs Wide Menu Structures

Arman Heydari

NSID: nki179

Interaction Problem

The domain of this study is user interaction with hierarchical menu systems, a common element in many digital interfaces. The problem arises when determining the optimal structure for these menus - specifically, whether a ‘wide’ menu with many items per level, or a ‘deep’ menu with many levels but fewer items, provides a better user experience. This becomes increasingly complex with the addition of numerous items and categories It’s tricky to find the right balance. Plus, as people use the system more, they get better at finding things, so we need to consider that too.

This study aims to explore these issues in depth, providing valuable insights for the design of more intuitive and user-friendly menu systems. We focused on the domain of a food-ordering menu application, where there are a variety of items and user want to select a single item in a menu. I’ll describe the details of this comparative analysis in the next section.

Interaction technique

I aimed to develop an application to compare the user’s performance on some tasks. The task is generally finding an item in a menu. But in different variations of the menu. We had about 5 people, from my friends, all with different levels of experience using computer menus, and different level of English proficiency to use our system. These users asked to search through menus to find specific items. There are 64 food items in overall, which I found in Internet and a Kaggle dataset, and categorize them in different levels.

Any user that has done the quiz, asked to find 10 distinct items each in a menu. Menus are different in branch factor and depth. Depth of a menu refers to the number of pages that user should select an option until they rich the intended item. Branch factor refers to number of options the user have in each page.

Dependent Variables: Average time that takes for users to find an item in the menu, number of wrong clicks (error) that they make until they click the intended item.

Independent variables: Depth of the menu, Branch factor of the menu.

Note that there are 64 items, so for instance when branch factor = 4 and depth = 2, here’s the number of options a user would encounter in that task to find the item: 4, 16. And in the case of branch factor = 2 and depth = 5, here’s the number of options a user would encounter in that task to find the item: 2, 2, 2, 2, 4.

Figure 1 shows an example of pages that a user might encounter. Where they should find an item called “Kona Hawaian coffe” in a menu. This is obviously a menu with branching factor 2 and depth 4. There are 10 tasks in each test that I got for a user.

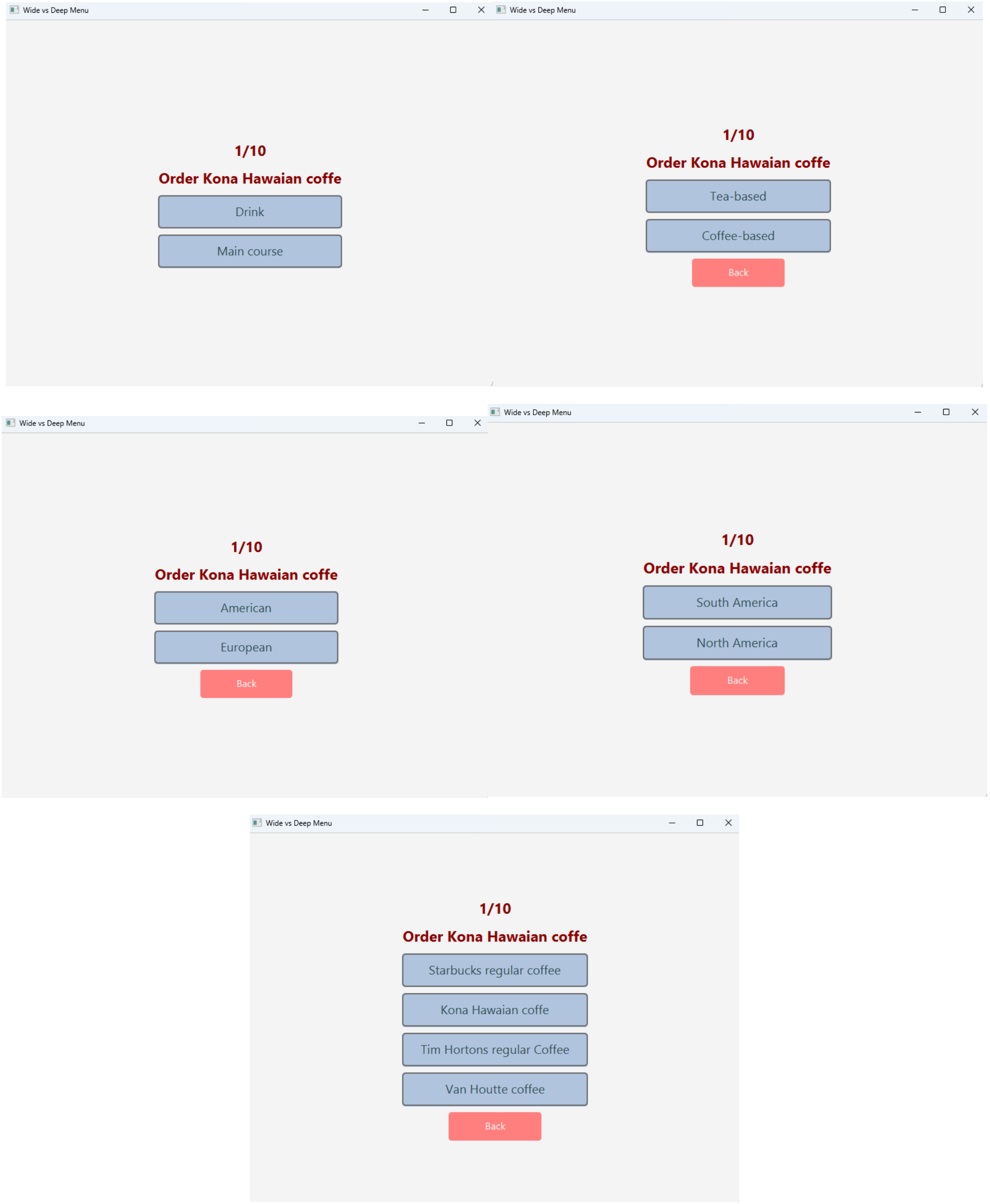


Figure 1: An example of pages in the application for a single task

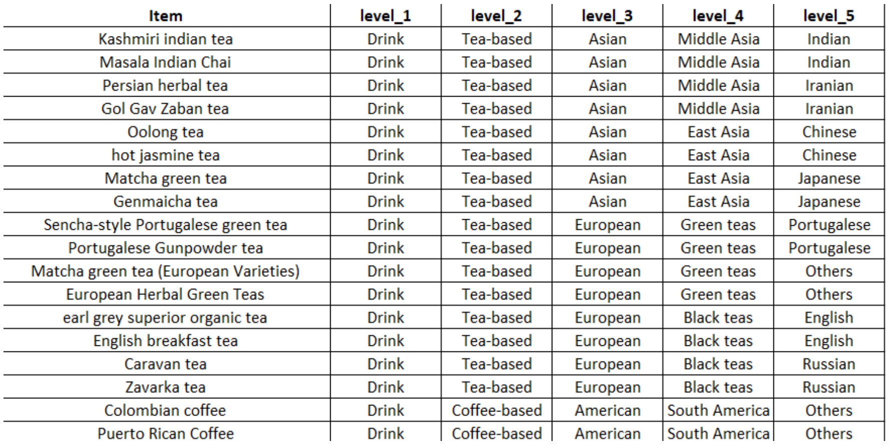
I focused on internal validity in this project. Because I only wanted to compare menus based on their width and not anything else. So, they all used my system's mouse and keyboard to answer the quiz. And the comparison is only on this dataset so we can't say the result is true for any data or application.

Technical Overview

I used Java as the programming language and JavaFX is a powerful tool for building interactive menus. It provides a rich set of UI components, including various menu types, with customization ability. I also used GitLab for version control. The hardware needed is just a standard computer with enough processing power to run Java and JavaFX.

The uploaded “Menu Analysis.zip” contains a javaFx application in a directory called Demo, “Menu.csv” file which is the source data of the application, “Output.csv” is the recorded results after tests, “analysis.ipynb” contains some brief python codes for analyzing results and plot charts, and the “Report Video.mp4” file is a video demonstrating the application capabilities.

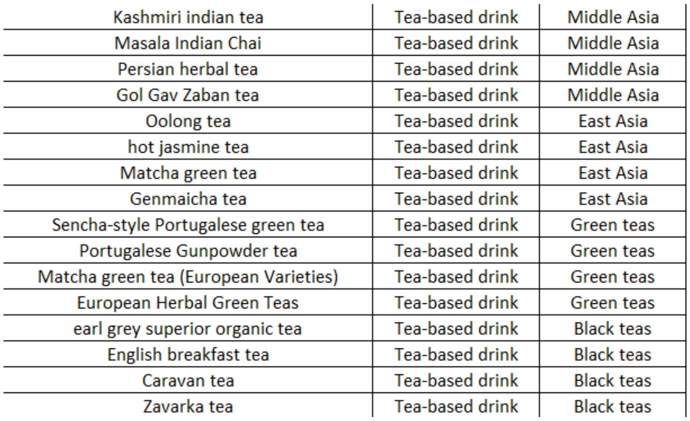
Menu.csv: Each of the first 64 items are the samples in tasks where branching factor is 4. As you can see in figure 2, They have 5 categories, and for any category we have equal number of sub-categories. This way, when finding an item with a menu of depth 4, the user has 2,2,2,8 options subsequently. Actually, there are always 2 choices which gives the user equal options after selecting any of them, unless it’s an item selection page which the number of items depends on the depth.



Item	level_1	level_2	level_3	level_4	level_5
Kashmiri indian tea	Drink	Tea-based	Asian	Middle Asia	Indian
Masala Indian Chai	Drink	Tea-based	Asian	Middle Asia	Indian
Persian herbal tea	Drink	Tea-based	Asian	Middle Asia	Iranian
Gol Gav Zaban tea	Drink	Tea-based	Asian	Middle Asia	Iranian
Oolong tea	Drink	Tea-based	Asian	East Asia	Chinese
hot jasmine tea	Drink	Tea-based	Asian	East Asia	Chinese
Matcha green tea	Drink	Tea-based	Asian	East Asia	Japanese
Genmaicha tea	Drink	Tea-based	Asian	East Asia	Japanese
Sencha-style Portugalese green tea	Drink	Tea-based	European	Green teas	Portugalese
Portugalese Gunpowder tea	Drink	Tea-based	European	Green teas	Portugalese
Matcha green tea (European Varieties)	Drink	Tea-based	European	Green teas	Others
European Herbal Green Teas	Drink	Tea-based	European	Green teas	Others
earl grey superior organic tea	Drink	Tea-based	European	Black teas	English
English breakfast tea	Drink	Tea-based	European	Black teas	English
Caravan tea	Drink	Tea-based	European	Black teas	Russian
Zavarka tea	Drink	Tea-based	European	Black teas	Russian
Colombian coffee	Drink	Coffee-based	American	South America	Others
Puerto Rican Coffee	Drink	Coffee-based	American	South America	Others

Figure 2: Dataset schema for branch factor 2

In the second 64 raw of this file, there are the same items but categorized in a way that for any category we have 4 equal number of sub-categories. So, there are always 4 choices which gives the user equal options after selecting any of them, unless it’s an item selection page which the number of items depends on the depth.



Kashmiri indian tea	Tea-based drink	Middle Asia
Masala Indian Chai	Tea-based drink	Middle Asia
Persian herbal tea	Tea-based drink	Middle Asia
Gol Gav Zaban tea	Tea-based drink	Middle Asia
Oolong tea	Tea-based drink	East Asia
hot jasmine tea	Tea-based drink	East Asia
Matcha green tea	Tea-based drink	East Asia
Genmaicha tea	Tea-based drink	East Asia
Sencha-style Portugalese green tea	Tea-based drink	Green teas
Portugalese Gunpowder tea	Tea-based drink	Green teas
Matcha green tea (European Varieties)	Tea-based drink	Green teas
European Herbal Green Teas	Tea-based drink	Green teas
earl grey superior organic tea	Tea-based drink	Black teas
English breakfast tea	Tea-based drink	Black teas
Caravan tea	Tea-based drink	Black teas
Zavarka tea	Tea-based drink	Black teas

Figure 3: Dataset schema for branch factor 4

JavaFX application: In the path “demo\src\main\java\com\example” There are 3 java files that is the main source code of the application. The other files in demo directory are created by jdk software when we initialize a javaFX application. I can’t explain the code line by line here but here’s an overview of important functionality:

- The App class is the main application class that extends Application. It sets up the primary stage and creates a new Page instance, which is the main component of the application.
- The Dataset class represents a dataset of menu items. It reads data from a CSV file and provides methods to select random items, filter lines based on a level and item, and save records to an output file.
- The Page class represents a page in the application. It displays a list of buttons representing menu items, a label showing the current order, and a label showing the remaining items.
- When a button is clicked, it checks if the answer is correct. If it is, it proceeds to the next page or saves the record if it's the last page. If the answer is incorrect, it increments the mistake count and proceeds to the next page.
- If it's not the first page, it also displays a back button to go to the previous page. Users need that specifically when they clicked a wrong option and realize it later.
- The display method in the Page class sets up the layout, creates the buttons and labels, and adds them to the scene. It also sets up event handlers for the buttons.

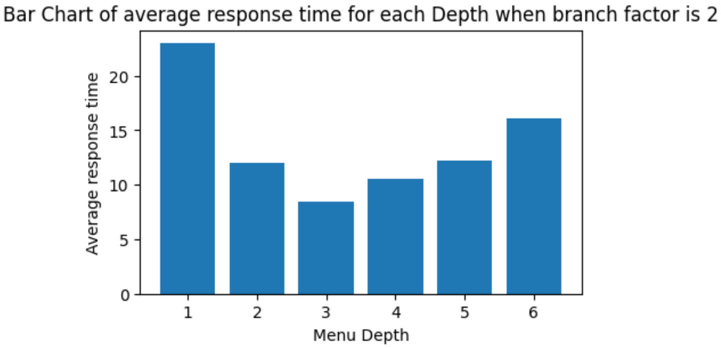
Evaluation

The main goal is to analyze users’ performance on different menus. By performance we mean the number of mistakes they make and the time it takes for them to find items. So we saved both of these values in the output. Evaluation methods are the branching factor and depth of menu that is tested, as described before. Table 1 shows how many tasks are there in each quiz with a setting. Note that depth=1 is the case that we have all the 64 items in a single page and it doesn’t different in branching factor 2 or 4, and we considered the results of that on both of the branching factor analysis.

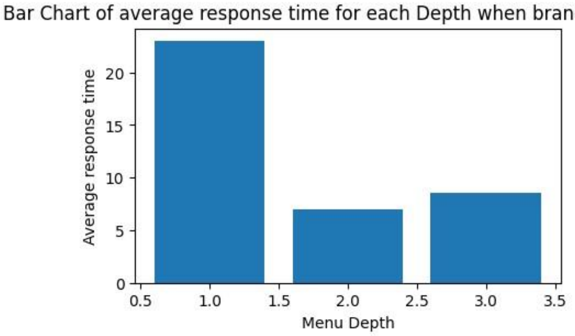
Table 1: Settings of variables in a 10 task quiz

	DEPTH=1	DEPTH=2	DEPTH=3	DEPTH=4	DEPTH=5	DEPTH=6
BF = 2	1	1	1	1	1	1
BF = 4	0	2	2	0	0	0

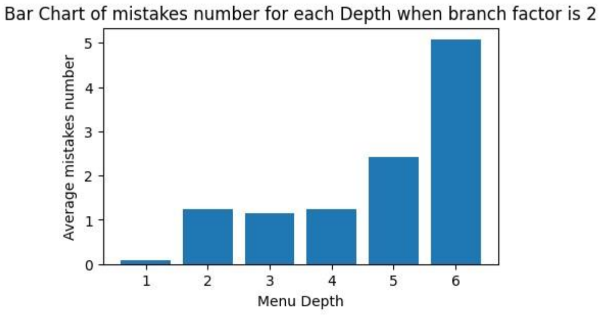
Know we want to see the results. The chart in following figure illustrates the average time our users took to locate items during the quiz, measured in seconds and plotted against the menu’s depth, when the branching factor is 2. With a very wide menu (depth=1), the visual search is time-consuming. At depth=2, the search time improves but still remains significant. Depth=3 appears to be the most efficient in our application, as users quickly make two general decisions before facing only 16 items. Depth=4 is similar for the same reasons. However, deeper menus, such as those with depth=5 or 6, seem less efficient because users spend more time making decisions and navigating through pages.

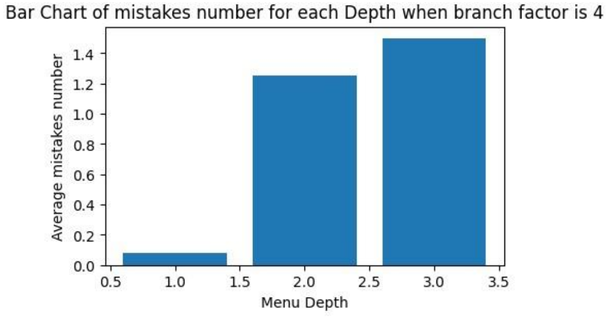


Also, when branching factor is 4, the most efficient menu is with depth=2, we can say for the same reason. With the difference that they limit the options more in their first select so it's better than depth=2 when branching factor was 2.

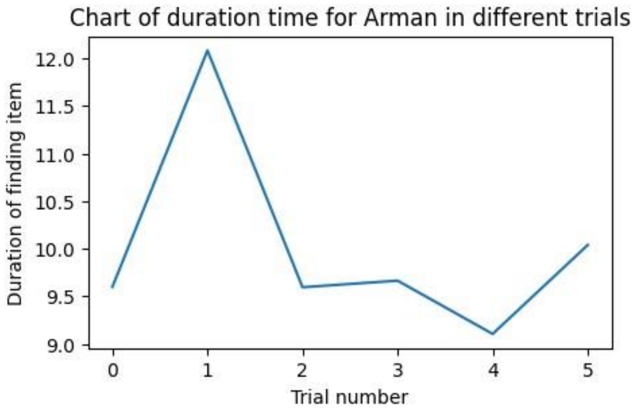


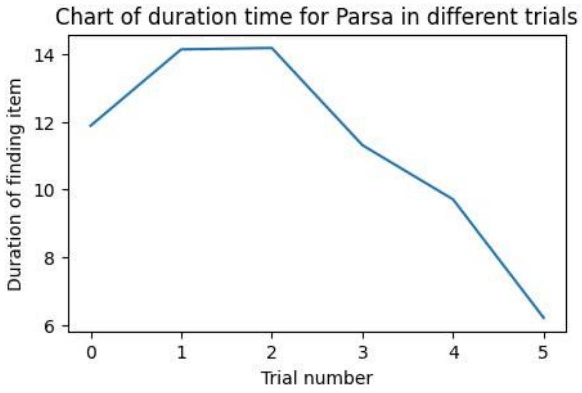
Furthermore, we can see in the charts in following figures, that users have done more mistakes when the menu is deep, and it doesn't different in branch factor 2 or 4. Which sound reasonable because they are facing lots of decisions for finding an item. It's true that in a deep menu they have less options each time and visual search is less time consuming, but on the other hand they have to make decisions about the category that they're looking for which causes more errors and they have to use the back button. Error is extremely higher in menu with depth 6.





Down figures demonstrate the effect of experience in results, where you can see each of the users' average responding time in finding items is generally decreased with the trials and they're negatively correlated. It only includes 2 of the users that answered the quiz more than once. In these figures the duration times is averaged over 5 consecutive tasks to prevent fluctuation and see the whole pattern.





Conclusion

In summary, this study aimed to investigate the optimal structure for hierarchical menu systems, specifically comparing 'wide' and 'deep' menus. The results show that the most efficient menu structure depends on the branching factor, with depth=3 being the most efficient for branching factor 2 and depth=2 being the most efficient for branching factor 4. Additionally, the results indicate that users make more mistakes in deeper menus, regardless of the branching factor. Furthermore, the study found that users' performance improves with experience, as they become more familiar with the menu structure. Based on these findings, system designers should consider the branching factor and depth of menus when designing hierarchical menu systems, and aim to find a balance between the two to optimize user performance. Because it seems there’s a trade-off between them. Additionally, designers should consider providing users with opportunities to practice and become familiar with the menu structure to improve their performance.