Optimizing Hierarchical Menus by Genetic Algorithm and Simulated Annealing

Shouichi Matsui
SERL, CRIEPI
2-11-1 Iwado-kita, Komae
Tokyo 201-8511, Japan
matsui@criepi.denken.or.jp

Seiji Yamada
National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda
Tokyo 101-8430, Japan
seiji@nii.ac.jp

ABSTRACT

Hierarchical menus are now ubiquitous. The performance of the menu depends on many factors: structure, layout, colors and so on. There has been extensive research on novel menus, but there has been little work on improving performance by optimizing the menu's structure. This paper proposes algorithms based on the genetic algorithm (GA) and the simulated annealing (SA) for optimizing the performance of menus. The algorithms aim to minimize the average selection time of menu items by considering the user's pointer movement and search/decision time. We will show the experimental results on a static hierarchical menu of a cellular phone as an example where a small screen and limited input device are assumed. We will also show performance comparison of the GA-based algorithm and the SA-based one by using wide varieties of usage patterns.

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation**]: User Interfaces — Interaction styles, Screen design.

General Terms

Human Factors, Design, Algorithms

Keywords

Hierarchical menu, optimization, genetic algorithm, simulated annealing.

1. INTRODUCTION

Hierarchical menus are one of the primary controls for issuing commands in GUIs. These menus have submenus as menu items and display submenus off to the side when they are selected as shown in Figure 1. The performance of the menu, i.e., the average selection time of menu items, depends on many factors, including its structure, layout, and colors. There have been many studies on novel menus (e.g., [2, 3, 7]), but there has been little work on improving the performance of a menu by changing its structure [1, 6, 11, 12]. Since a very simple search method gave a fairly good improvement [1], we proposed an algorithm based on the genetic algorithm (GA) for minimizing the average selection time [12].

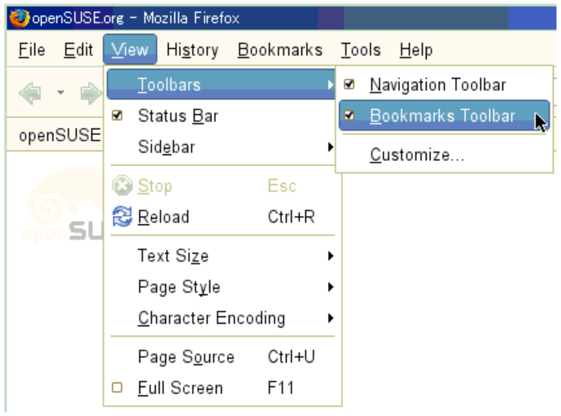

Figure 1: Example of hierarchical menu for a web browser.

There have been many studies on menu-design and menu-layout from the standpoint of the user interface. Francis et al. were the first to optimize a multi-function display that is very similar to a hierarchical menu using the simulated annealing (SA) [6]. Quiroz et al. proposed an interactive evolution of a non-hierarchical menu using an interactive evolutionary computation (IEC) approach [14].

Liu et al. applied a visual search model to menu design [11]. They used the Guided Search (GS) model to develop menu designs. They defined a GS simulation model for a menu search task, and estimated the model parameters that would provide the best fit between model predictions and experimental data. Then they used an optimization algorithm to identify the menu design that minimized the predicted search times according to predefined search frequencies of different menu items, and they tested the design. Their results indicate that the GS model has the potential to be part of a system for predicting or automating the design of menus.

Amant et al. showed the concepts to support the analysis of cellular phone menu hierarchies [1]. They proposed a model-based evaluation of cellular phone menu interaction, gathered data and evaluated three models: Fitts' law model, GOMS, and ACT-R. They concluded that the prediction by GOMS was the best among the three models. They also tried to improve menu selection time by using a simple best-first search algorithm and got over 30% savings in selection time.

This paper proposes algorithms based on the genetic algorithm (GA) and the simulated annealing (SA) for optimizing the performance of menus. The algorithms aim to minimize the average selection time of menu items by considering the user's pointer movements and search/decision time.

We will show the experimental results on a static hierarchical menu of a cellular phone as an example where a small screen and limited input device are assumed. We will also show performance comparison of GA-based algorithm and the SA-based one by using wide variety of the usage patterns.

2. FORMULATION OF THE PROBLEM

2.1 Overview

The optimization problem of hierarchical menus can be considered as one dealing with placing menu items on the nodes of a tree. Let us assume a tree where the maximum depth is D, the maximum number of children that a node has is W, the root is the initial state, and menu items are on nodes. An example of a hierarchical menu is shown in Figure 2. As shown in the figure, some menu items have children; i.e. some menu items have submenus. The time to select the target item is the time to traverse from the root to the target node. The problem is to minimize the average traversal time with respect to the given search frequencies of menu items.

We cannot arbitrarily arrange the menu purely for efficiency. We must respect the semantic relationships between the items. That is, "Ringer Volume" is under the "Settings" category rather than vice versa for good reason. To cope with the difficulties of representing and reasoning about menu item semantics we introduce two metrics: *functional similarity* and *menu granularity*.

Functional similarity is a metric that represents the similarity of two menu items in terms of their functions. We assume that the functional similarity takes a value between 0 and 1; 0 means that the two items have no similarity and 1 means that the two items have very high similarity. For example it is very natural to assume that "Create New Mail" and "Favorite Web Site" have low similarity and that "Create New Mail" and "Inbox of Mail" have high similarity. We use this metric to avoid placing items with low similarity on the same submenu of a node. If items with low similarity are put on the same submenu, then it becomes difficult for a user to recognize and memorize the menu layout. The formal definition will be given later.

Menu granularity is a metric that reflects the number of submenus a node has as its descendants. We introduce this metric to avoid placing an item that has many children and an item that has no child as children of the same node. The formal definition will be given later.

The problem of minimizing the average traversal time is a very difficult one because of the following constraints;

- The traversal time from a node to its children is not constant; it varies depending on the starting and ending nodes.
- Menu items usually belong to groups, and they have hierarchical constraints.
- We should consider the functional similarity and the menu granularity of each item from the standpoint of usability.

2.2 Formulation

Let l be the level number, i is the ordering number in siblings, and v_i^l be a node of a tree (Figure 2). Moreover, let $M = (V, E)$ be a tree where $V = \{v_i^l\}$ denotes the nodes, and $E = \{e_{ij}\}$

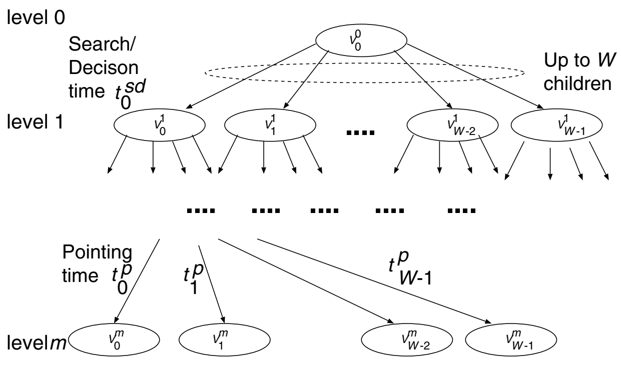

Figure 2: Tree structure of a hierarchical menu.

denotes the edges. We call the leaf nodes that correspond to generic functions "terminal nodes."

There are two kinds of menu item or node in M. One type is terminal nodes that correspond to generic functions, and the other is intermediate nodes. The terminal nodes cannot have children.

Let I_i represent a menu item and the total number of items be N; i.e., there are $I_i (i = 1, \cdots, N)$ menu items. Items that correspond to generic functions are less than N, and some items/nodes are intermediate items/nodes that have submenu(s) as a child or children. We assume that a menu item I_i is assigned to a node v_i^l, therefore we use I_i and v_i^l interchangeably.

We also assume that the selection probability of the terminal node/generic function is represented by Pr_i.

2.2.1 Selection Time

The selection time t_i^l of a menu item/node v_i^l on the hierarchical level l can be expressed using the search/decision time t_i^{sd} and the pointing time t_i^p as $t_i^l = t_i^{sd} + t_i^p$ [5]. We also consider the time to reach level l; therefore, the whole selection time T_i of a node v_i^l on level l can be expressed as $T_i = \sum_{j=0}^{l-1} t_{i_j}^j + t_i^l$. Thus the average selection time T_{avg} is as follows:

$$T_{avg} = \sum_{i=1}^{N} Pr_i T_i. \tag{1}$$

2.2.2 Pointing Time

As Silfverberg et al. [15] and Cockburn [5] reported, the pointing time t_i^p can be expressed by using Fitts' law as $t_i^p = a^p + b^p \log_2(A_i/W_i + 1)$, where the coefficients a^p and b^p are determined empirically by regressing the observed pointing time, A_i is the distance moved, and W_i is the width of the target.

Fitts' law describes the time taken to acquire, or point to, a visual target. It is based on the amount of information that a person must transmit through his/her motor system to move to an item – small, distant targets demand more information than large close ones, and consequently they take longer to acquire. Therefore, the term $\log_2(A_i/W_i + 1)$ is called the *index of difficulty (ID)*.

2.2.3 Search/Decision Time

We assume that the search/ decision time t_i^{sd} can be expressed as follows [5].

- For an unexperienced user, the time required for a linear search is $t_i^{sd} = b^{sd} n^l + a^{sd}$, where n^l is the number of items that a level l node has, and the coefficients a^{sd} and b^{sd} are determined empirically by regressing the observed search time.

- For an expert, we can assume that the time t_i^{sd} obeys Hick-Hyman's law and can be expressed as. $t_i^{sd} = b^{sd}H_i + a^{sd}$, $H_i = \log_2(1/Pr_i^l)$, where the coefficients a^{sd} and b^{sd} are determined empirically by regressing the observed search time. If we can assume that all items are equally probable, $H = \log_2(n^l)$ iff $\forall Pr_i^l = 1/n^l$.

2.2.4 *Functional Similarity*

Toms et al. reported the result of generating a menu hierarchy from functional descriptions using cluster analysis [17]. However, this approach is time consuming therefore, we chose to use another one. We represent the functional similarity of item I_x and I_y by using a function $s(I_x, I_y)$ which takes a value between 0 and 1. Let us assume that a generic function of each item I_i can be specified by some words $wl_i = \{w_0, w_1, \cdots\}$, and let $\mathbf{WL} = \bigcup_i wl_i$ be the whole words. Let us also assume that an intermediate node can be characterized by the words by which the children are specified. Let $\mathbf{x}$ be a vector in which element x_i represents the frequency of the i-th word in its specification, and let $\mathbf{y}$ be a vector of node y. Then, the functional similarity $s(I_x, I_y)$ is defined as $s(I_x, I_y) = \dfrac{\mathbf{x}\cdot\mathbf{y}}{|\mathbf{x}||\mathbf{y}|}$. This similarity is widely utilized in information retrieval field [4].

Let us consider a node v_i^l that has m children. The penalty of functional similarity $P^s_{v_i^l}$ of node v_i^l is defined as $P^s_{v_i^l} = \sum_{x=0}^{m-1}\sum_{y=i+1}^{m-1} (1 - s(I_x, I_y))$. And the total penalty P^s is defined as follows:

$$P^s = \sum_{v_i^l \in \{V \setminus v_0^0\}} P^s_{v_i^l}. \qquad (2)$$

2.2.5 *Menu Granularity*

The menu granularity $g_{v_i^l}$ of a node v_i^l is defined as the total number of descendants. If node v_i^l is a terminal node, then $g_{v_i^l} = 0$. Moreover, if node v_i^l has m children $(v_j^{l+1}, j = 0, \cdots, m-1)$ whose menu granularities are $g_{v_j^{l+1}}, (j = 0, \cdots, m-1)$, then $g_{v_i^l}$ is defined as $g_{v_i^l} = \sum_{j=0}^{m-1} g_{v_j^{l+1}}$. The penalty of menu granularity $P^g_{v_i^l}$ of node v_i^l is defined as $P^g_{v_i^l} = \sum_{i=0}^{m-1}\sum_{j=i+1}^{m-1}\left|g_{v_i^l} - g_{v_j^l}\right|$. And the total penalty P^g is defined as follows:

$$P^g = \sum_{v_i^l \in \{V \setminus v_0^0\}} P^g_{v_i^l}. \qquad (3)$$

2.2.6 *Objective Function*

The problem is to minimize the following objective function:

$$f = T_{avg} + \alpha P^s + \beta P^g, \qquad (4)$$

where α and β are constants that control the preference of functional similarity and menu granularity.

2.3 Local/Partial Optimization

2.3.1 *Placing Items as Children of a Node*

Let us consider a node v_i^l on level l that has $n \le W$ children $v_j^{l+1}(j = 0, \cdots, n-1)$, and represent the traversal time from v_i^l to v_j^{l+1}, i.e., the pointing time for v_j^{l+1} by t_j^l. When we want to place $I_j, (j = 0, \cdots, n-1)$ menu items whose selection probabilities are represented by Pr_j as the children of the v_i^l, the average pointing time $T_{v_i^l}$,

$$T_{v_i^l} = \sum_{j=0}^{n-1} Pr_j t_j^l, \qquad (5)$$

is minimized as follows:

1. Sort I_i using Pr_i as the sort key in descending order, and let the result be $I_i'(i = 0, \cdots, n-1)$,
2. Sort v_i^{l+1} using t_i^l as the sort key in ascending order, and let the results be $v_i^{\prime(l+1)}(i = 0, \cdots, n-1)$
3. Placing I_i' on the node $v_i^{\prime(l+1)}$ gives the minimum average pointing time from node v_i^l.

2.3.2 *Optimization Problem*

When menu items that are placed as the children of a node V are given, the placement that minimizes the average pointing time is straightforward. Therefore, the problem is to find the best assignment of menu items to nodes of a tree that minimizes Equation (4), where nodes have a fixed capacity of W items. There should be at least $L = \lceil N/W \rceil$ nodes in the tree and N items placed on some node. The first node has W items chosen from N items, and the second node has W items chosen from $N - W$ items, and so on, so the search space of the problem is roughly ${}_NC_W \times {}_{N-W}C_W \times \cdots \times {}_{N-LW}C_W = N!/(W!)^L$, therefore the problem is a difficult combinatorial optimization problem. For instance, consider the case of $N = 200$, $W = 10$. The search space is roughly $200!/((10!)^{20}) \sim 10^{243}$.

3. ALGORITHMS BASED ON GA AND SA

3.1 Basic Strategy

Previous studies showed that breadth was preferable to depth [9, 10, 16, 18, 19]. Schultz and Curran reported that menu breadth was preferable to depth [16]. Larson and Czerwinski reported the results of depth and breadth tradeoff issues in the design of GUIs [10]. Their results showed that, while increased depth did harm search performance on the web, a medium condition of depth and breadth outperformed the broadest shallow web structure overall.

Zaphiris studied the effect of depth and breadth in the arrangement of web link hierarchies on user preference, response time, and errors [18]. He showed that previous menu depth/breath tradeoff procedures were applicable to the web link domain. He also showed that task completion time increased as the depth of the web site structure increased. Zaphiris et al. also showed the results of the study investigating age related differences as they relate to the depth versus breadth tradeoff in hierarchical online information systems [19]. They showed that shallow hierarchies were preferred to deep hierarchies, and seniors were slower but did not make more errors than their younger counterparts when browsing web pages.

Because the previous studies showed that breadth was preferable to depth, we use a kind of breadth-first search algorithm (shown later) as the core of the proposed GA.

3.2 GA-based Algorithm

3.2.1 *Chromosome and Mapping from Genotype to Phenotype*

A simple way to represent a solution of the problem is a tree. But there is a problem that genetic operators such as crossover or mutation may generate an infeasible solution; i.e., the tree does not contain all the generic functions. There are two ways to cope with this problem. The first way is to convert an infeasible solution into a feasible one, and also modify the chromosome. The other way is to use a chromosome representation that does not generate infeasible solutions. We based the proposed algorithm on the latter approach.

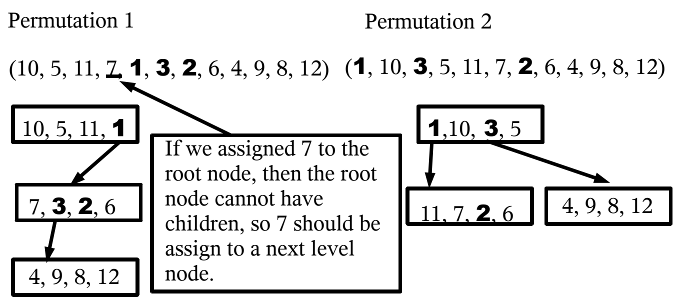

Figure 3: Mapping from permutation to a tree structure.

Since breadth is preferable to depth, an algorithm that places menu items I_i one by one on a usable node that has the smallest node number can find a good solution. We number each node from root to bottom, and from left to right. We use an algorithm that assigns I_i to a node as follows:

1. A chromosome of the GA is a sequence of I_i, i.e., a chromosome can be represented as a permutation of numbers.
2. According to the permutation, assign menu items I_i one by one to vacant positions of the node that has the smallest node number.
3. If a generic function is assigned to a node, then the number of children that the node can have is decreased by 1.

If we have a sufficient number of intermediate nodes, we can search enough space to find the optimal solution.

Two examples of assignment according to permutation are depicted in Figure 3, where W is 4. In the figure, numbers in bold **(1, 2, 3)** represent the intermediate node. Let us consider "Permutation 1". In this case, we can assign "10", "5", and "11" to the root node. But we cannot assign "7" to the root node, because the root node cannot have any children if we did. Therefore, we should assign "7" to the next level node, and the remaining position of the root node should be an intermediate node. Because there is an intermediate node in the root node, we can assign "**1**" to the root node.

In the case of "Permutation 2", the mapping is straightforward. The first number "**1**" is an intermediate node, so we assign it to the root node, and the number of vacant positions in the tree is incremented by 4. The next number "10" can be assigned to the root node, and "**3**" and "5" can be assigned to the root node. The remaining numbers are assigned to the children of the root nodes.

3.2.2 *Crossover and Mutation*

We used a crossover operator that does not generate an invalid chromosome. As described above, a chromosome is a permutation of numbers; therefore, we used crossover operators that are developed for the representation. We used the CX (Cycle Crossover) [13], OX (Order Crossover) [13], PMX (Partially-Mapped Crossover) [13] for the crossover operators and compared the performance.

We use the swap mutation as the mutation operator. Randomly chosen genes at position p and q are swapped.

The crossover and mutation operators do not generate invalid chromosomes; i.e., offspring are always valid permutations.

3.2.3 *Other GA Parameters*

The selection of the GA is tournament selection of size 2. The initial population is generated by random initialization; i.e., a chromosome is a random permutation of numbers. We used a steady state GA, for which the population size is 100, and the mutation rate was one swap per chromosome.

```
STEP1: $T \leftarrow T_{max}$
    select a permutation $V_c$ at random
STEP2: pick a point $V_n$ from the neighborhood of $V_c$
    if eval($V_n$) is better than eval($V_c$)
        then select it ($V_c \leftarrow V_n$)
        else select it with probability $e^{-\frac{\Delta eval}{T}}$
    repeat this step $k_T$ times
STEP3: set $T \leftarrow \gamma T$
    if $T \geq T_{min}$ then goto STEP2 else goto STEP1
```

Figure 4: Outline of SA-based algorithm.

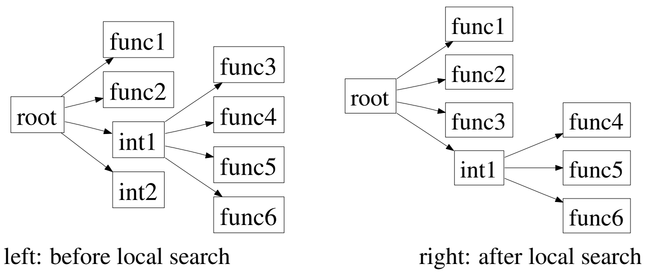

Figure 5: Local search.

3.3 SA-based Algorithm

We also developed an algorithm based on the simulated annealing (SA). The outline of the algorithm that is adopted from the book by Michalewicz and Fogel [13] is shown in Figure 4. We used the same representation and mapping of the GA; therefore, V_c and V_n in Figure 4 represent permutations of numbers that correspond to I_i. We also used the same local search in the SA-based algorithm. The neighborhood is generated by using the swap mutation operator described above.

We used the following settings based on the preliminary experiments:

Initial temperature: $T_{max} = 10$
Minimum temperature: $T_{min} = 0.001$
Cooling ratio: $\gamma = 0.9$
Cooling interval: $k_T = 100$

3.4 Local Search

We used a local search method to improve the performance of GA and SA. The method finds an unused node v_i^l; i.e., finds an intermediate node that has no child, and swaps v_i^l with a node that is the sibling's child v_j^{l+1} when swapping the two decreases the average selection time. Figure 5 shows an example of this procedure. In the left tree, the intermediate node "int2" has no child, so it is swapped with "func3", and the result is the right part.

4. EXPERIMENTS

We conducted experiments to confirm the effectiveness of the proposed algorithm, to compare the performance of GA-based and SA-based algorithms, to compare the performance of different crossover operators CX, OX, and PMX.

The target was a cellular phone that is used by one of the authors. The phone [8] has 24 keys as shown in Figure 6. The target phone has hardware keys for "E-mail", "EZweb", "Phone book",

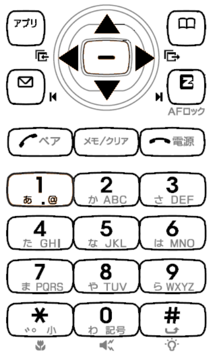

Figure 6: Key layout of the target cellular phone.

and "Application". Also there is a "Shortcut"key (cursor down). The root menu thus has the four submenus corresponding to the hardware keys.

4.1 Experimental Data

4.1.1 Pointing Time and Search/Decision Time

The index of difficulty for 24×24 key pairs were calculated as follows. We measured the relative coordinates of the center (x, y) of each key and measured width and height of each key. We calculated the index of difficulty to an accuracy of one digit after the decimal point. This gave us 28 groups of indices of difficulty.

We measured the pointing time of one-handed thumb users for the above 28 groups by recording the tone produced by each key press [1]. Unpaid volunteers participated in the experiment. We prepared 28 tasks corresponding to the 28 groups. We got $t_i^p = 192 + 63\log_2(A_i/W_i + 1)$ (ms) for predicting the pointing time, and the equation is very similar to the one reported by Silfvergerg et al. [15][1] Although the target phone has the ability to select a menu item by pressing a key that is prefixed to item title, we assumed that all selections were done by cursor movements.

The target of this experiments was an expert; therefore, we used $t_i^{sd} = 80\log_2(n^l) + 240$ (ms) [5][2];

4.1.2 Usage Frequency Data and Similarity

We gathered the usage frequency data as follows. The first author recorded the daily usage of each function for two months, and we generated the usage frequency data from the record. There are 129 terminal nodes in the data. We call the data "Original."

We also generated the following dataset from the "Original." We assumed a user who frequently used e-mail application (Mail2, Mail3) and a user who frequently used Web application (Web2, Web3).

Mail2 The frequencies of E-mail related items are multiplied by 2.

Mail3 The frequencies of E-mail related items are multiplied by 3.

Web2 The frequencies of Web related items are multiplied by 2.

Web3 The frequencies of Web related items are multiplied by 3.

We assigned three to five words to each generic function according to the users' manual of the target phone [8].

[1] $t_i^p = 176 + 64\log_2(A_i/W_i + 1)$ (ms).
[2] The equation is derived from experiments conducted for a computer display, and is not for a cellular phone.

4.2 Results for Original Frequency Data

We conducted the following experiments.

case 1 Typical Usage: This experiment was conducted to assess the typical improvement by the GA and SA. The maximum width W was 16.

case 2 Limited Breadth: Although breadth is preferable to depth, pressing "Down" key many times is sometimes tedious. This experiment was conducted to see the effect of limiting the breadth. In this case, we set W to 12, 9, and 6.

Because GA and SA are stochastic algorithms, we conducted 50 runs for every test case, and the results shown in Table 1 are averages over 50 runs. The two parameters for weights were set to $\alpha = 10.0$ and $\beta = 1.0$. The maximum number of fitness evaluations was 100,000.

4.2.1 Performance Comparison

Table 1 shows the results (T_{ave}) of 4 algorithms, GA-based ones with CX, OX, PMX, and SA-based one. Figures 7 and 8 show the results of average selection time and the values of objective function in box-plot format.

In Table 1, "Local Move" shows the results of a local modification that places menu items according to their frequency, i.e., the most frequently used item is placed as the top item, and so on. "Imp(%)" means the improvement from the original menus that is defined as $Imp = (T_{orig} - T_{opt})/T_{orig} \times 100$, where T_{orig} is the average traversal time of the original menu and T_{opt} is the average traversal time of optimized menu. The bold faces mean the best among the algorithms. The "ave" means the average over 50 runs, "max" means the maximum and "min" means the minimum among 50 runs. As the table shows, the proposed algorithms can generate menu with shorter average selection times. Moreover, limiting the number of usable keys gave us better menus. This is partly because the search/decision time is proportional to $\log_2(n)$, where n is the number of items. As the number of items increases, the search/decision time increase; therefore, the average selection time increase. Limiting the number of keys to 6 gave longer selection times.

From Table 1, Figure 7, and Figure 8 we can conclude as follows:

- The SA-based algorithm performed best from the stand point of average performance for $W = 12, 9, 6$. The GA-based one using PMX performed best for $W = 16$. The SA-based algorithm found menus with smaller average value of objective function in all cases. It also found the minimum in all cases.

- The variance of objective function is large in GA-based algorithms when compared with the SA-based one. There is no significant difference among the GA-based algorithms.

- The GA-based algorithm using CX found a menu structure with the minimum average selection time for $W = 12, 6$. The SA-based one found the best for $W = 16$ and the the GA-based one using OX found the best for $W = 9$.

The original menu (T_{ave}=3331 (ms)) and the best menu of Case 2 ($W = 9$), with the minimum objective function, (T_{ave}=1953 (ms)) that was found by the GA-based algorithm with OX are shown in Figure 11. In the figure, items and intermediate nodes are shown in boxes and the vertical ordering shows the placement in a single level menu. The box is omitted for low usage frequency items/ intermediate nodes for the sake of saving space.

Table 1: Improvements in Average Selection Time.

	CX				OX				PMX				SA			
	T_{ave}(ms)		Imp.(%)		T_{ave}(ms)		Imp.(%)		T_{ave}(ms)		Imp.(%)		T_{ave}(ms)		Imp.(%)	
Case	ave	min	ave	max	ave	min	ave	max	ave	min	ave	max	ave	min	ave	max
Original	3331		0.0		3331		0.0		3331		0.0		3331		0.0	
Local Move	2812		15.0		2812		15.0		2812		15.0		2812		15.0	
Case 1 (W=16)	2036	1949	38.9	41.5	2032	1972	39.0	40.8	**2029**	1979	**39.1**	40.6	2032	**1917**	39.0	**42.4**
Case 2 (W=12)	1980	**1886**	40.6	**43.4**	1985	1918	40.4	42.4	1970	1930	40.9	42.1	**1967**	1931	**41.0**	42.0
Case 2 (W=9)	1949	1915	41.5	42.5	1948	**1887**	41.5	**43.4**	1956	1924	41.3	42.2	**1947**	1893	**41.5**	43.2
Case 2 (W=6)	2233	**2004**	33.0	**39.9**	2245	2015	32.6	39.5	2223	2015	33.3	39.5	**2217**	2014	**33.4**	39.5

Table 2: Effect of Weights.

		CX				OX				PMX				SA			
weight		T_{ave}(ms)		Imp.(%)		T_{ave}(ms)		Imp.(%)		T_{ave}(ms)		Imp.(%)		T_{ave}(ms)		Imp.(%)	
α	β	ave	min	ave	max	ave	min	ave	max	ave	min	ave	max	ave	min	ave	max
0	0	1823	1793	45.3	46.2	1819	**1749**	45.4	**47.5**	1820	1788	45.4	46.3	**1797**	1784	**46.1**	46.4
5	1	1919	**1871**	42.4	**43.8**	1918	1873	42.4	43.8	**1917**	1876	**42.4**	43.7	1925	1885	42.2	43.4
20	1	2013	1925	39.6	42.2	2017	**1921**	39.4	**42.3**	1998	1935	40.0	41.9	**1997**	1922	**40.1**	42.3
40	1	2072	**1916**	37.8	**42.5**	2053	1939	38.4	41.8	**1949**	1920	**41.5**	42.3	2041	1922	38.7	42.3
20	5	2020	1940	39.4	41.8	2013	1927	39.6	42.1	2015	1927	39.5	42.1	**2011**	**1901**	**39.6**	**42.9**
20	10	2005	**1902**	39.8	**42.9**	2002	1910	39.9	42.7	**1995**	1917	**40.1**	42.5	2011	1923	39.6	42.3

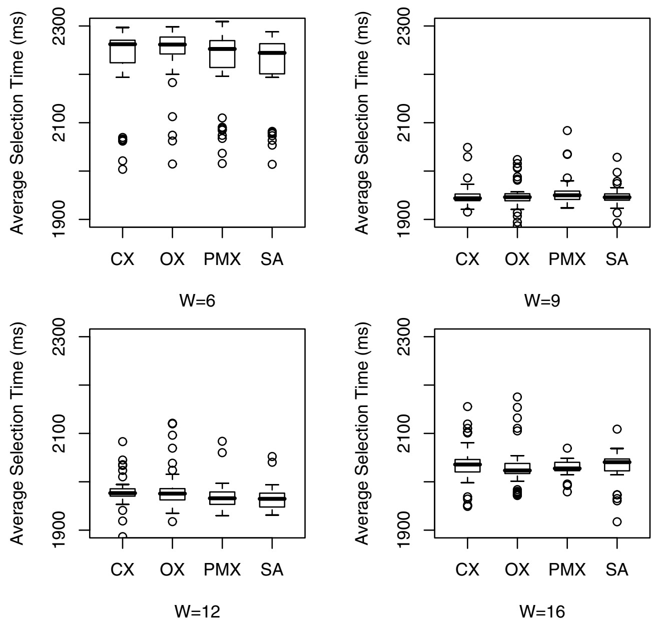

Figure 7: Comparison of average selection time (Original).

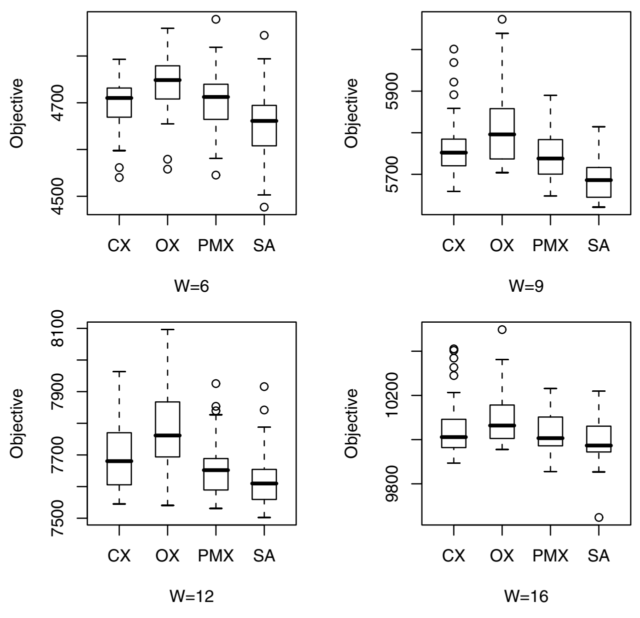

Figure 8: Comparison of objective function (Original).

In Figure 11, items with high usage frequency are placed on a smaller level and on an upper position. For example, the most frequently used "Inbox folder 2", which is placed under the "Inbox" menu in the original menu, is placed as a child of "E-Mail" in the optimized menu. Note also that "Shortcut" is not used in the original menu, but it is fully utilized in the optimized menu; frequently used URLs are placed in "Shortcut".

4.2.2 *Effects of Weights*

We introduced two weights for the penalties of functional similarity and of menu granularity. Table 2 shows the results of different weight settings for the case $W = 9$. Table 2 shows that the average selection time increased as we increased α. The effect of β is not the same, the average selection time did not always increased as we increased β. This is partly because there is the constraint of maximum width W.

The SA-based algorithm performed well for wide varieties of

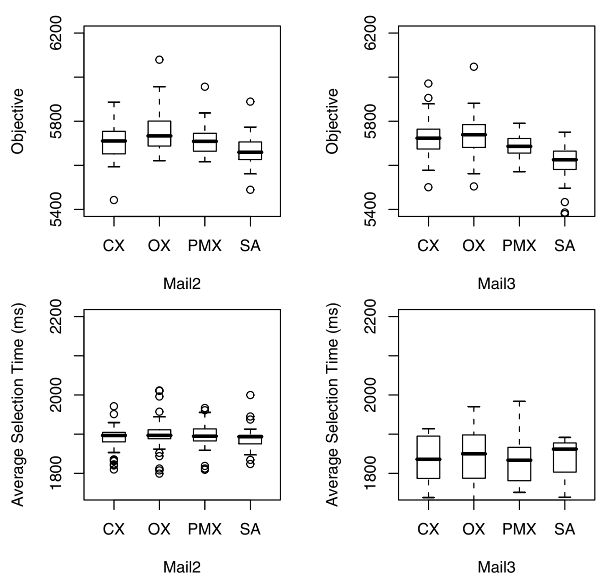

Figure 9: Performance comparison (Mail2, Mail3).

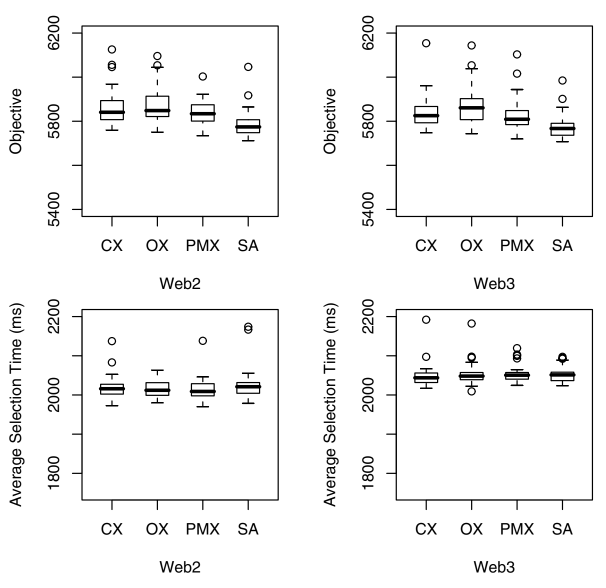

Figure 10: Performance comparison (Web2, Web3).

weight settings from the stand point of average performance. However, from the stand point of minimum average selection time, the GA-based one with CX or OX performed better.

4.3 Results for Other Dataset

Figures 9 and 10 depict the results of the average selection time and the values of objective function in box-plot format for the "Mail2", "Mail3", "Web2", and "Web3" datasets. We can conclude as follows:

- The SA-based algorithm performed well for the four datasets from the stand point of average performance. It found menus with smaller objective function.
- The GA-based algorithm found a menu with the minimum average selection time for "Mail2" and "Mail3" datasets.

5. DISCUSSION AND FUTURE WORK

The experiments show that the proposed algorithms can generate better menu hierarchies for the target phone. The proposed 4 algorithms found menus with nearly 40% reduction in the average selection time. The SA-based algorithm performed best from the standpoint of average performance.

Our model has four parameters (a^p, b^p,a^{sd}, b^{sd}) that are determined empirically by regressing the observed data, and the average selection time depends these parameters. However, we confirmed that we could get nearly same improvements for wide ranges of the parameters' settings.

Because the target of the proposed algorithm is not limited to cellular phones, and the preliminary results are promising, we will apply the algorithm to wider varieties of targets including other hierarchical menus in office application and tree-structured data base like bookmark of Web browser. In this paper we only focused on the static menu as the target; adaptive/dynamic menu (e.g., [2,3,7]) that changes menu contents in adapting to varying usage is a future target.

The data that were used in the experiments, especially selection frequency data, are limited, therefore it is also a future work to gather wider variety of usage data, and confirm the effectiveness of the proposed method.

We used CX, OX, and PMX as the crossover operators and there were no significant performance difference among them and we might have to develop a new crossover operator to improve the performance of GA-based algorithm.

6. CONCLUSION

We proposed GA-based and SA-based algorithms for minimizing the average selection time of menu items that consider the user's pointer movement time and the decision time. The results showed that the algorithms can generate a better menu structure. They found over 40% reduction in the average selection time for wide variety of usage patterns. The target of the proposed algorithms is not limited to cellular phones, and the algorithms can be applied to other hierarchical menus.

7. REFERENCES

[1] St. Amant, T.E. Horton, and F.E. Ritter. Model-based evaluation of cell phone menu interaction. In *Proc. CHI 2004*, pages 343–350 2004.

[2] D. Ahlström. Modeling and improving selection in cascading pull-down menus using Fitts' law, the steering law and force fields. In *Proc. CHI 2005*, pages 61–70, 2005.

[3] J. Beck, S.H. Han, and J. Park. Presenting a submenu window for menu search on a cellular phone. *Int. J. of Human-Computer Interaction*, 20(3):233–245, 2006.

[4] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*, Addison-Wesley,1999.

[5] A. Cockburn, G. Gutwin, and S. Greenberg A predictive model of menu performance. In *Proc. CHI 2007*, pages 627–636, 2007.

[6] G. Francis. Designing multifunction displays: an optimization approach. *Int. J. of Cognitive Ergonomics*, 4(2):107–124, 2000.

[7] L. Findlater and J. McGrenere. A comparison of static, adaptive, and adaptable menus. In *Proc. CHI 2004*, pages 89–96, 2004.

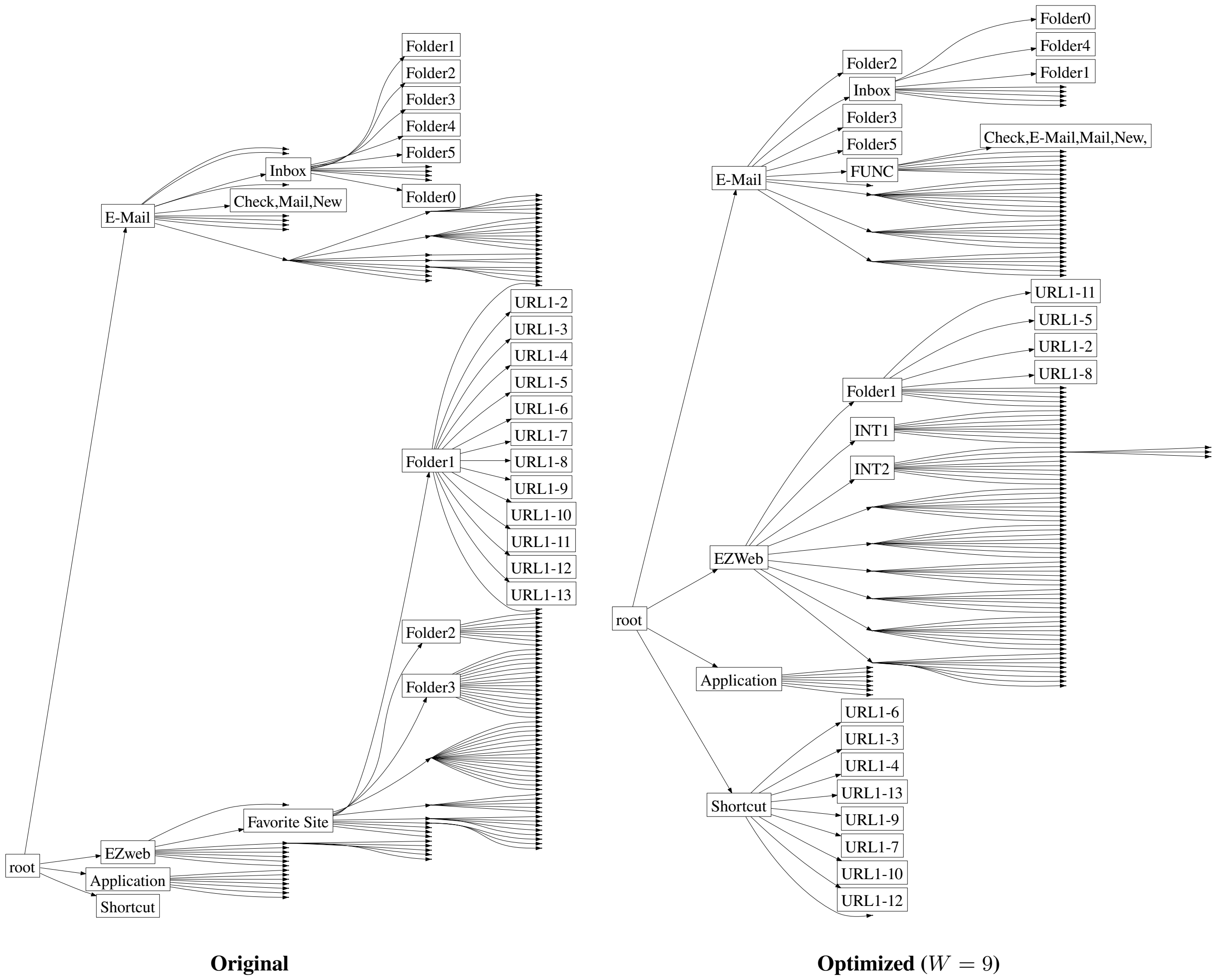

Figure 11: Comparison of menus.

[8] KDDI: Manual for CASIO W43CA, http://www.au.kddi.com/torisetsu/pdf/w43ca/w43ca_torisetsu.pdf, 2006.

[9] J. I. Kiger. The depth/breadth trade-off in the design of menu-driven user interfaces. *Int. J. Man-Mach. Stud.*, 20(2):201–213, 1984.

[10] K. Larson and M. Czerwinski. Web page design: implication of memory, structure and scent for information retrieval. In *Proc. CHI 1998*, pages 25–32, 1998.

[11] B. Liu, G. Francis, and G. Salvendy. Applying models of visual search to menu design. *Int. J. Human-Computer Studies*, 56:307–330, 2002.

[12] S. Matsui and S. Yamada. Genetic algorithm can optimize hierarchical menus, In *Proc. of CHI 2008*, 2008 (*to appear*).

[13] Z. Michalewicz and D.B. Fogel. *How to solve it: modern heuristics*, Springer-Verlag, 2000.

[14] J.C. Quiroz, S.J. Louis, and S. M. Dascalu. Interactive evolution of XUL user interfaces. In *Proc. of GECCO 2007*, pages 2151–2158, 2007.

[15] M. Silfverberg, I.S. MacKenzie, and T. Kauppinen. Predicting text entry speed on mobile phones. In *Proc. CHI 2000*, pages 9–16, 2000.

[16] E.E. Schultz Jr. and P.S. Curran. Menu structure and ordering of menu selection: independent or interactive effects? *SIGCHI Bull.*, 18(2):69–71, 1986.

[17] M.L. Toms, M.A. Cummings-Hill, D.G. Curry, and S.M. Cone. Using cluster analysis for deriving menu structures for automotive mobile multimedia applications. SAE Technical Paper Series 2001-01-0359, SAE, 2001.

[18] P. Zaphiris. Depth vs breadth in the arrangement of web links. In *Proc. 44th Annual Meeting of the Human Factors and Ergonomics Society*, pages 139–144, 2000.

[19] P. Zaphiris, S.H. Kurniawan, and R.D. Ellis. Age related difference and the depth vs. breadth tradeoffs in hierarchical online information systems. In *Proc. User Interfaces for All, LNCS 2615*, pages 23–42, 2003.

[20] M. Ziefle and S. Bay. Mental models of a cellular phone menu. Comparing older and younger novice users. In *Proc. MobileHCI 2004, LNCS 3160*, pages 25–37, 2004.