A Genetic Algorithm for Optimizing Hierarchical Menus

Shouichi Matsui and Seiji Yamada

Abstract— **Hierarchical menus are widely used as a standard user interface in modern applications that use GUIs. 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 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 useage patterns.**

I. 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. Cellular phones that have only small displays show submenus as new menus, as shown in Fig. 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 improving the performance of a menu by changing its structure [1], [5], [6], [11], [12]. A very simple search method gave a fairly good improvement [1]; therefore, we can expect further performance improvements by optimizing the structure.

There have been many studies on menu design, menu layout from the standpoint of the user interface. Francis et al. were the first to optimize a multi-function display that was essentially the same as a hierarchical menu by using Simulated Annealing (SA) [5], [6]. Quiroz et al. proposed an interactive evolution of a non-hierarchical menu using an interactive evolutionary computation (IEC) approach.

Liu et al. applied a visual search model of 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.

S. Matsui is with the System Engineering Research Laboratory (SERL), Central Research Institute of Electric Power Industry (CRIEPI), 2-11-1 Iwado-kita, Komae, Tokyo 201-8511, Japan; email: matsui[at]criepi.denken.or.jp.

S. Yamada is with the National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda, Tokyo 101-8430, Japan; email: seiji[at]nii.ac.jp.

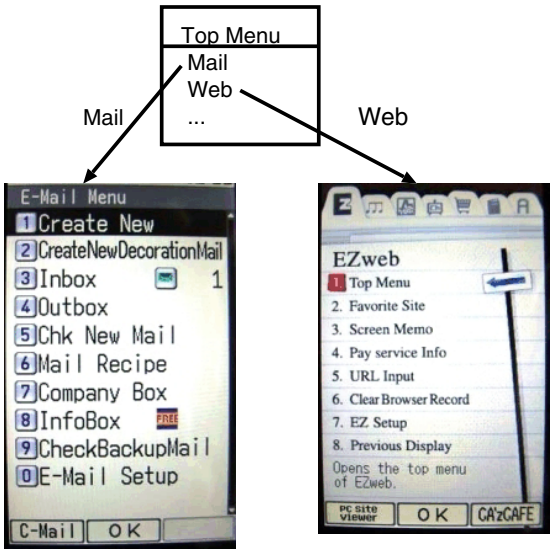


Fig. 1. Example of Hierarchical Menu for a Cellular Phone.

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 an algorithm based on the genetic algorithm (GA) for optimizing the performance of menus. The algorithm aims to minimize the average selection time of menu items by considering the user's pointer movement and search/decision time.

We will show preliminary results on a static hierarchical menu of a cellular phone as an example for a device with a small screen and limited input capability.

II. Formulation of the Problem

A. 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 Fig. 2.

978-1-4244-1823-7/08/$25.00©2008 IEEE

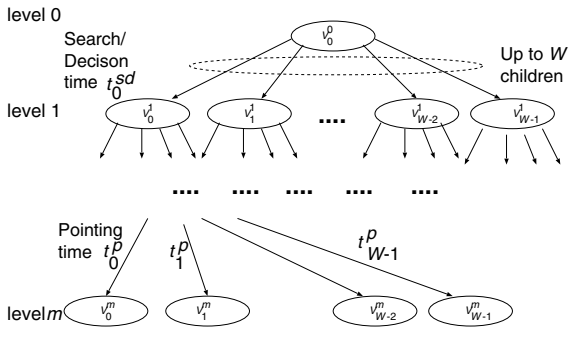


Fig. 2. Tree Structure of a Hierarchical Menu

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, it becomes difficult for a user to remember 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.

B. Formulation

1) Notation: Let l be the level number, i is the ordering number in siblings, and v_i^l be a node of a tree (Fig. 2). Moreover, let $M = (V, E)$ be a tree where $V = \{v_i^l\}$ denotes the nodes and $E = \{e_{ij}\}$ 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) 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 follows [4]:

$$t_i^l = t_i^{sd} + t_i^p. \tag{1}$$

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 follows:

$$T_i = \sum_{j=0}^{l-1} t_{i_j}^j + t_i^l. \tag{2}$$

Thus, the average selection time T_{avg} is defined as follows:

$$T_{avg} = \sum_{i=1}^{N} Pr_i T_i. \tag{3}$$

3) Pointing Time: As Silfverberg et al. [13] and Cockburn [4] reported, the pointing time t_i^p can be expressed by using the Fitts' law as follows:

$$t_i^p = a + b \log_2(A_i/W_i + 1), \tag{4}$$

where the coefficients a and b 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)*,

4) Search/Decision Time: We assume that the search/ decision time t_i^{sd} can be expressed as follows [4].

- For an inexperienced user, the time required for a linear search is as follows:

$$t_i^{sd} = b^{sd} n^l + a^{sd}, \tag{5}$$

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.

$$t_i^{sd} = b^{sd} H_i + a^{sd}, \tag{6}$$

$$H_i = \log_2(1/Pr_i^l), \tag{7}$$

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

$$H = \log_2(n^l) \qquad \text{iff} \quad \forall Pr_i^l = 1/n^l. \tag{8}$$

5) Functional Similarity: Toms et al. reported the result of generating a menu hierarchy from functional descriptions using cluster analysis [15]. However, this approach is time consuming; therefore, we choose 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 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 follows:

$$s(I_x, I_y) = \frac{\mathbf{x} \cdot \mathbf{y}}{|\mathbf{x}||\mathbf{y}|} \tag{9}$$

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 follows:

$$P^s_{v_i^l} = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} (1 - s(I_i, I_j)). \tag{10}$$

And the total penalty P^s is defined as follows:

$$P^s = \sum_{v_i^l \in \{V \setminus v_0^0\}} P^s_{v_i^l}. \tag{11}$$

where S^{int} represents the set of intermediate nodes.

6) 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 follows:

$$g_{v_i^l} = \sum_{j=0}^{m-1} g_{v_j^{l+1}}. \tag{12}$$

The penalty of menu granularity $P^g_{v_i^l}$ of node v_i^l is defined as follows:

$$P^g_{v_i^l} = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} \left| g_{v_i^l} - g_{v_j^l} \right|. \tag{13}$$

And the total penalty P^g is defined as follows:

$$P^g = \sum_{v_i^l \in \{V \setminus v_0^0\}} P^g_{v_i^l}. \tag{14}$$

where S^{int} represents the set of intermediate nodes.

7) Objective Function: The problem is to minimize the following objective function:

$$f = T_{avg} + \alpha P^s + \beta P^g, \tag{15}$$

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

C. Local/Partial Optimization

1) Placing Items as Children of a Node: Let us consider a node v_i^l on level l that has $n \leq W$ 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 $T_{v_i^l}$,

$$T_{v_i^l} = \sum_{j=0}^{n-1} Pr_j t_j^l, \tag{16}$$

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 $v_i^{'(l+1)} (i = 0, \cdots, n-1)$
3) Placing I_i' on the node $v_i^{'(l+1)}$ gives the minimum average pointing time from node v_i^l.

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 (15), 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 $200!/((10!)^{20}) \sim 10^{243}$.

III. Genetic Algorithm

A. Basic Strategy

Previous studies showed that breadth was preferable to depth [9], [10], [14], [16], [17]. Schultz and Curran reported that menu breadth was preferable to depth [14]. 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 [16]. He showed that previous menu depth/breath tradeoff procedures applied 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 [17]. 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.

B. 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 modify the chromosome. The other way is to use a chromosome representation that does not generate infeasible solutions. We base the proposed algorithm on the latter approach.

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.

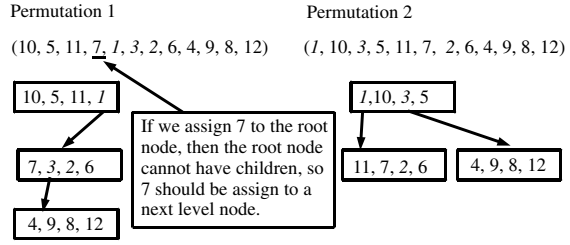
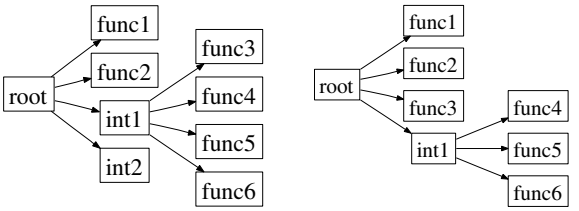


Fig. 3. Mapping a Permutation to a Tree Structure.



left: before local search right: after local search

Fig. 4. Local Search.

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 Fig. 3, where W is 4. In the figure, numbers in italic 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.

C. Local Search

We use a local search method to improve the performance of GA. 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}. Figure 4 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.

D. Crossover and Mutation

We use a crossover operator that does not generate an invalid chromosome. As described above, a chromosome is a permutation of numbers; therefore, we use crossover operators that are developed for the representation. Based on

Fig. 5. Key Layout of the Target Cellular Phone.

the results of preliminary experiments, we chose CX (Cycle Crossover) for the crossover operator .

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.

E. 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 use a steady state GA, for which the population size is 100, and the mutation rate is one swap per chromosome.

IV. Numerical Experiments

We conducted numerical experiments to confirm the effectiveness of the proposed algorithm. The target was a cellular phone that is used by one of the authors. The phone [8] has 24 keys as shown in Fig. 5.

The target phone has hardware keys for "E-mail", "EZweb", "Phone book", and "Application". And there is a "Shortcut"key (cursor down). The root menu thus has the four submenus corresponding to the the hardware keys.

A. Experimental Data

1) Pointing Time and Decision Time: The index of difficulty for 24×24 key pairs was calculated as follows. We measured the relative coordinates of the center (x, y) of each key and measured the 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 indexes of difficulty as shown in Table I. We named each key, from top to bottom and left to right, as follows: "App", "Up", "Phone book", "Left", "Center", "Right", "Mail", "Down", "Web", "Call", "Clear", "Power", "1", thru "9", "*", "0", and "#".

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]. There are two ways to measure the pointing time. Silfverberg et al. measured the time by counting the number of characters generated by key presses in 10 seconds [13]. Amant et al. measured the time by recording the tone produced by each key press [1]. Because the target has keys that do not generate any character, such as cursor keys, we measured the time by recording the tone.

TABLE I
Index of Difficulty for the Target Phone (24 keys)

group #	ID	example pairs		# of pairs
		from	to	
1	3.7	*	Up	2
2	3.6	0	Up	3
3	3.5	9	Up	6
4	3.4	8	Up	8
5	3.3	8	Right	17
6	3.2	9	Down	22
7	3.1	8	Down	25
8	3.0	6	Right	28
9	2.9	1	Up	29
10	2.8	8	Center	29
11	2.7	1	*	33
12	2.6	2	Right	29
13	2.5	1	9	29
14	2.4	1	0	53
15	2.3	1	3	33
16	2.2	2	Center	20
17	2.1	1	8	25
18	2.0	2	Call	17
19	1.9	1	7	21
20	1.8	Mail	Call	7
21	1.7	1	5	50
22	1.6	1	2	16
23	1.4	2	Clear	9
24	1.3	Right	Up	12
25	1.2	1	4	21
26	1.1	Center	Down	4
27	0.8	Right	Center	4
28	0.0	1	1	24

Unpaid volunteers participated in the experiment. We prepared 28 tasks corresponding to the 28 groups. The "Read Email Message" function of the phone was used during the tasks, except for the one task (ID=1.4, "2" to "Clear"). For the exceptional case, the "write memo" function (with number mode selected) was used. The participants repeated the task of pressing the "From" key and the "To" key 10 times for each task. The pointing time was calculated by subtracting the starting time of the tone of "From" from the starting time of tone of "To."

We got the following equation for predicting the pointing time, and the equation is very similar to the one reported by Silfvergerg et al. [13][1]

$$t_i^p = 192 + 63 \log_2(A_i/W_i + 1) \quad (\text{ms}). \tag{17}$$

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 experiment was an expert; therefore, we used the following equation for the search/decision time [4][2]:

$$t_i^{sd} = 80 \log_2(n^l) + 240 \ (\text{ms}). \tag{18}$$

[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.

TABLE II

IMPROVEMENT IN AVERAGE SELECTION TIME.

Case	T_{ave}(ms)	(%)	P^s	P^g
Original	3331	0.0	454	793
Local Move	2812	15.5	454	793
Case 1 (W=16)	2036	38.9	727	1259
Case 2 (W=12)	1998	40.0	541	856
Case 2 (W=9)	1959	41.2	402	291
Case 2 (W=6)	2237	32.8	279	173

TABLE III

EFFECT OF WEIGHTS.

α	β	T_{ave}(ms)	(%)	P^s	P^g
0.0	0.0	1837	44.9	584	448
5.0	1.0	1935	41.9	405	278
10.0	1.0	1959	41.2	402	291
20.0	1.0	1990	40.3	396	300
40.0	1.0	2066	38.0	395	309
20.0	5.0	2011	39.6	397	274
20.0	10.0	2028	39.1	405	260

2) Usage Frequency Data: We gathered 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 were 129 terminal nodes in the data.

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

B. Results

We conducted the following experiments.

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

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

Because GA is a stochastic algorithm, we conducted 50 runs for every test case, and the results shown in Table II and Table III 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.

In Table II, "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. As the table shows, the proposed algorithm can generate menu with shorter average selection time. Moreover, limiting the breadth give 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 increases; therefore, the average selection time increases. Limiting the breadth to 6 gave a longer selection time and smaller penalties.

The original menu (T_{ave}=3331 (ms)) and the best menu of Case 2 (9 keys) (T_{ave}=1913 (ms)) are shown in Fig. 7 and Fig. 8. In the two figures, 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.

In Fig. 8, 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".

C. Effects of weights

We introduced two weights for the penalties of functional similarity and of menu granularity. Table III shows the results of different weights settings for the case $W = 9$. The average selection time increased as we increased α. The table also shows that the difference in average selection time was larger than that of the penalty factor of P^s. Setting them to zero gave a shorter selection time, but the penalties were larger.

There is a tradeoff among the average selection time, functional similarity, and menu granularity; therefore, a multi-objective approach might be a more natural formulation.

D. Convergence Speed

Figure 6 shows fitness, average selection time, and two penalty terms in the best case ($W = 9$). GA found a fairly good solution within 50,000 fitness evaluations. The penalty term of "Functional Similarity" decreased almost monotonically, but the term of "Menu Granularity" oscillated in the early stage. The average selection time initially decreased rapidly, but sometimes increased in the middle of iteration because of the penalty terms.

V. DISCUSSION AND FUTURE WORK

The experiments show that the proposed algorithm can generate better menu hierarchies for the target phone. Because our targets of are not limited to cellular phones, and the preliminary results are promising, we will apply the algorithm to wider varieties of targets such as Web browser bookmarks.

In this paper, we focused on a static menu as the target; adaptive/dynamic menu (e.g., [2], [3], [7]) that changes menu contents depending on usage will be a future target.

The data used in the experiments, especially selection frequency data, were limited. Therefore, we should gather a wider variety of usage data and use that to confirm the effectiveness of the proposed method.

VI. CONCLUSION

We proposed a GA-based algorithm for minimizing the average selection time of menu items that considers the user's pointer movement time and the decision time. The preliminary results showed that the algorithm can generate a better menu structure. The target of the proposed algorithm is not limited to cellular phones.

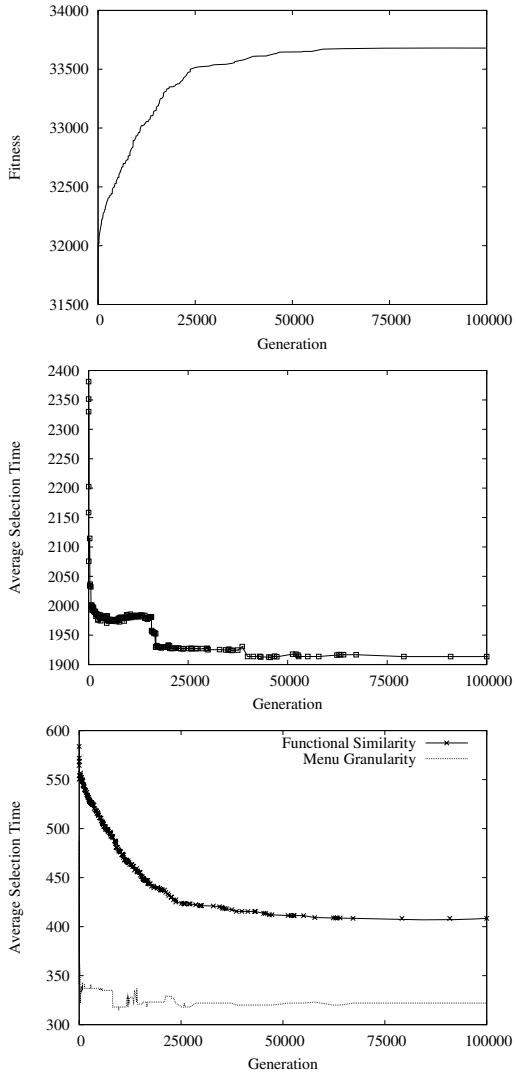


Fig. 6. Fitness, Average selection time, Penalty terms.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Fujio Tsutsumi of CRIEPI for his valuable comments.

REFERENCES

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

[2] D. Ahlström,"Modeling and improving selection in cascading pull-down menus using Fitts' law, the steering law and force fields," *Proc. CHI 2005*, pp.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*, vol.20, no.3, pp.233–245, 2006.

[4] A. Cockburn, G. Gutwin, and S. Greenberg, "A predictive model of menu performance," *Proc. CHI 2007*, pp.627–636, 2007.

[5] G. Francis, "Designing multifunction displays: an optimization approach," *Int. J. of Cognitive Ergonomics*, vol.4, no.2, pp.107–124, 2000.

[6] G. Francis and C. Rash, "MFDTool(version 1.3): a software tool for optimizing hierarchical information on multifunction displays," USAARL Report No.2002-22, August 2002.

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

[8] KDDI: "Manual for CASIO W43CA," http://www.au.kddi.com/torisetsu/pdf/w43ca/w43ca_torisetsu.pdf, http://www.au.kddi.com/torisetsu/pdf/w43ca/w43ca_kantan.pdf, 2006.

[9] J. I. Kiger, "The depth/breadth trade-off in the design of menu-driven user interfaces," *Int. J. Man-Mach. Stud.*, vol.20, no.2, pp.201–213,1984.

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

[11] B. Liu, G. Francis, and G. Salvendy, " Applying models of visual search to menu design," *Int. J. Human-Computer Studies*, no.56, pp.307–330, 2002.

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

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

[14] E.E. Schultz Jr. and P.S. Curran, "Menu structure and ordering of menu selection: independent or interactive effects?," *SIGCHI Bull.*,vol.18, no.2, pp.69–71,1986.

[15] 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.

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

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

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

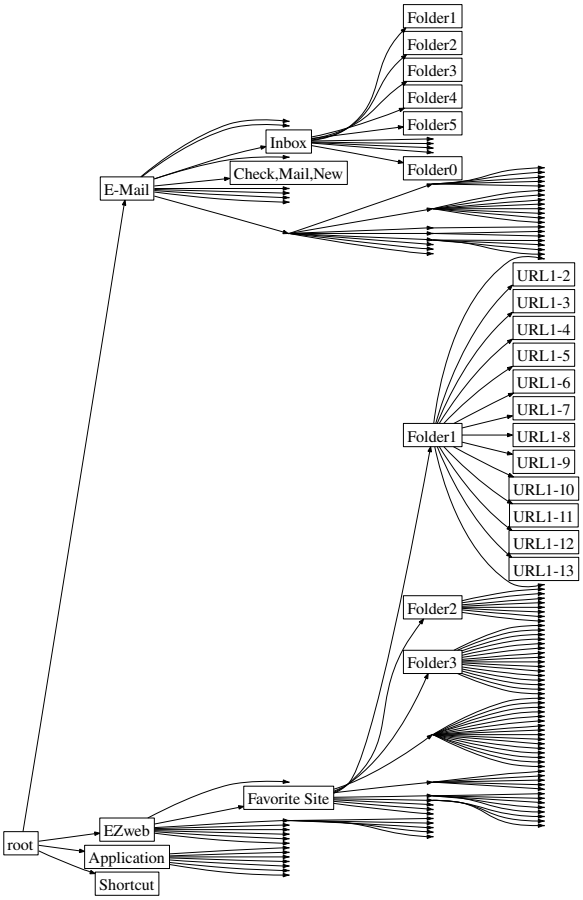


Fig. 7. Original Hierarchical Menu.

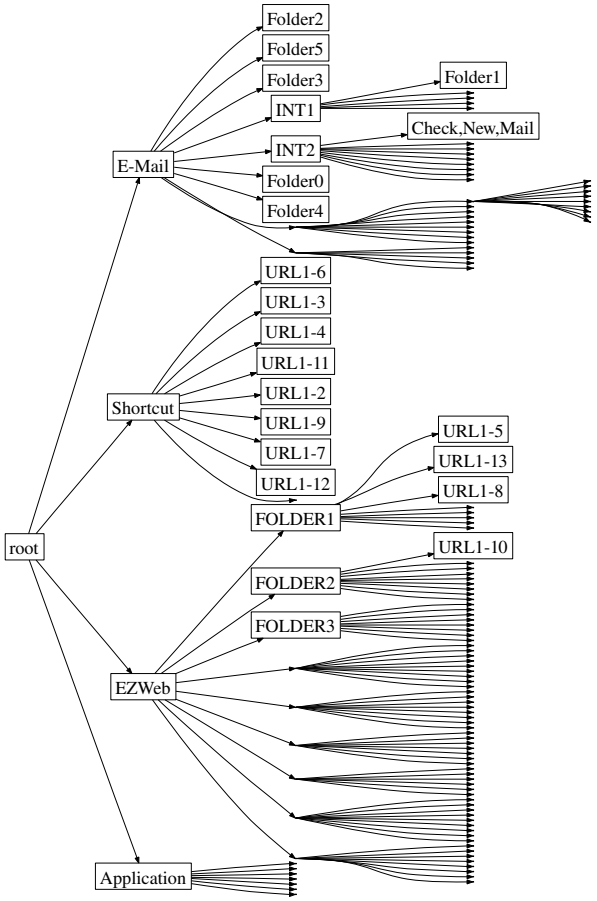


Fig. 8. Example of an Optimized Menu ($W = 9$).