ユーザの適応を考慮した階層メニューの最適化

Optimization of Hierarchical Menus Considering User Adaptation

松井 正一*1　山田 誠二*2
Shouichi Matsui　Seiji Yamada

*1(財) 電力中央研究所
Central Research Institute of Electric Power Industry

*2国立情報学研究所／総合研究大学院大学
National Institute of Informatics, SOKENDAI

Hierarchical menus are now ubiquitous. The performance of the menu depends on many factors: structure, layout, colors and so on. We have proposed 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 movement and decision time. But the algorithm does not explicitly consider user adaptation to new menu. We propose a new formulation for optimization of hierarchical menu that includes a model of user adaptaton to new menu.

1. はじめに

階層メニューは GUI でコマンドを指定する用途などで多用されている．階層メニューの性能は構造，レイアウト，色などの多くの項目によって決まる．現在までに，ユーザインタフェースの観点から多くの研究が行われ，様々なメニューの方式が提案されているが [Ahlström 05]，構造を変更することで性能（目的の項目に辿り着くまでの時間が短い）を向上する研究は少ない．Amant らは携帯電話のメニューを対象に，単純な最適化手法により選択時間を削減能している [Amant 07].

我々は遺伝的アルゴリズムによる階層メニューの最適化方式を提案し，携帯電話と PDA を例に提案手法の有効性を示した [松井 08].

既提案手法では，メニュー構造を変更した後に必要なユーザ適応については考慮していない．これに対し，本報告では，ユーザ適応まで考慮した最適化手法を提案する．

2. 問題の定式化

2.1 既提案での定式化の概要

既提案手法 [松井 08] の定式化の概要を以下に示す.

l で木構造の階層番号を，i で子の順番を，v_i^l でノードを表す．機能に対応するノードを「終端ノード」と呼び、サブメニューを持つノードを「中間ノード」と呼ぶ．機能 i の頻度は選択確率 P_i で与えられるものとする．メニュー項目を I_i で表現し，その総数は N，つまり, $I_i (i = 1, \cdots, N)$ とする．

階層メニューの最適化は木構造のノードにメニュー項目を適切に配置する問題として考えることができる．最大の深さが D で 1 ノードが最大 W の子を持つ木構造を考え，根が初期状態に相当し，メニュー項目はノードに配置されるものとする．中間項目に相当するノードは子としてサブメニューを持つ．目的の項目を選択するために必要な時間は，根から目的のノードまでの到達時間となる．最適化の目的は，各項目毎に与えらる利用頻度の下で，平均選択時間を最小化することである．

使いやすさの観点からは，効率だけを考えて任意の項目を任意の場所に配置することは望ましくなく，項目の意味を尊重する必要がある．このために,「機能の類似度」と「メニューの粒度」という二つの尺度を導入する．

連絡先: 松井 正一，(財) 電力中央研究所 システム技術研究所，201-8511 狛江市岩戸北 2-11-1，matsui [at] criepi.denken.or.jp

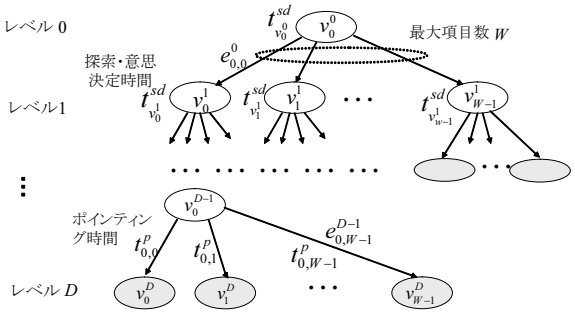

図 1: 階層メニューの木構造表現

2.1.1 選択時間

階層メニュー上のレベル l にあるノード，メニュー項目 v_i^l の選択時間 t_i^l は探索・意思決定時間 t_i^{sd} とポインティング時間 t_i^p を用いて $t_i^l = t_i^{sd} + t_i^p$ と表現できる [Cockburn 07]．最終的な項目を選択するためには，根からレベル l まで辿り着く必要があり，レベル l にあるノード v_i^l の選択時間 T_i は根から目的ノードまでの総和となる．したがって平均選択時間 T_{avg} は $T_{avg} = \sum_{i=1}^{N} P_i T_i$ で表せる．

2.1.2 ポインティング時間

ポインティング時間 t_i^p は Fitts の法則を使って $t_i^p = a^p + b^p \log_2(A_i/W_i + 1)$ で表現できる [Cockburn 07] ．ここで，$\log_2(A_i/W_i + 1)$ は困難度の指標 (Index of difficulty) と呼ばれる．係数 a^p, b^p は実験データの回帰式で決定する．

2.1.3 探索・意思決定時間

レベル l にある n^l 個の項目を持つノードでの探索・意思決定時間 t_i^{sd} は以下と仮定する [Cockburn 07].

- 初心者については $t_i^{sd} = b^{sd} n^l + a^{sd}$ とする．
- 熟練者に対しては，Hick-Hyman 法則が成り立つものとし，$H_i = \log_2(1/P_i^l)$ として，$t_i^{sd} = b^{sd} H_i + a^{sd}$ を仮定する．選択確率は一定とすれば，項目数を n として $H_i = \log_2(n)$ である．

それぞれの式の係数は実験データの回帰式で決定する．

2.1.4 目的関数

機能の類似度のペナルティを P^s，メニューの粒度のペナルティを P^g として，目的関数は次式となる．$f = T_{avg} + \alpha P^s + \beta P^g$，ここで α と β は P^s と P^g をどの程度重要視するかを調整するためのパラメータである．

この目的関数を最小化する問題は複雑な組み合わせ最適化問題となることから，遺伝的アルゴリズムを用いた解法を提案した [松井 08].

3. ユーザ適応への拡張

3.1 基本的な考え方

ユーザ適応について考える場合には，以下の考慮が必要である．ユーザに依存して変化する属性は以下である．

1. 各メニュー項目の利用頻度分布.
2. ユーザの適応（習熟）による探索・意思決定時間の短縮，それによる各項目への到達時間の短縮.

また，ユーザの予測とユーザビリティのトレードオフをうまく扱うことが重要である [Gajos 08]．システムがユーザに適応することとユーザがシステムに適応することのトレードオフを考えると，基本的には下記の戦略が有望と考えられる．

1. 最小限の変化で最大限の効果：できるだけ現在のメニュー構成を変更せずに，変更後の効果ができるだけ大きくなる評価関数が望ましい．これは予測に合致する適応の方が好まれるためである [Gajos 08].
2. 評価関数は，メニュー変更前後のパフォーマンスの差分と，変更された部分にユーザが適応するコストの見積もりから構成する．
   (a) 変更前のパフォーマンス：$P = M - (T_{avg} + \alpha P^s + \beta P^g)$
   (b) 変更後のパフォーマンス：$P' = M - (T'_{avg} + \alpha P^{s\prime} + \beta P^{g\prime})$
   (c) メニュー変更へのユーザの適応コスト AC

ここで，M は前述の目的関数 f が最小化する形のものであることから，パフォーマンスの最大化とするために導入する定数である．基本的には，$F = (P' - P) - \gamma AC$ を最大化する問題として定式化する．ここで γ は適応コストをどの程度重視するかの予め定める係数である．

3.2 ユーザの習熟のモデル化

ユーザは，変更前のメニューには習熟していると仮定できるため，探索・意思決定時間には熟練者のモデルを用い，メニューが変更された直後は未習熟であることから，初心者のモデルを用いる．変更後のメニューの利用を継続することで，ユーザは習熟していくことから，ある期間を経過した後は，熟練者のモデルを用いることが妥当と考えられる．

Cockburn らは [Cockburn 07]，初心者の場合の探索・意思決定時間を T_{vsi}，熟練者の場合を T_{hhi}，習熟度を e_i として，探索・意思時間は $t^{sd} = (1 - e_i)T_{vsi} + e_i T_{hhi}, (0 \leq e_i \leq 1)$ で表現できるとしている．習熟度は L を learnability のパラメータ，s_i を試行回数として，$e_i = L \times (1 - 1/s_i)$ で表現される．また実験によりその妥当性を示している．

L はメニュー選択に要する平均的な移動距離をメニュー項目数から推定するとしており，試行の度にメニューの配置が変わる場合には，$L = 0$ となる．配置が固定であれば，利用頻度から平均的な移動距離は計算できる．

3.3 ユーザの適応コスト

従来の習熟に関する実験 [Cockburn 07] は，新しいメニューに対する習熟に関してであり，古いメニューからの移行を考えたものではない．これに対して，提案手法では，習熟したメニューから新しいメニューに適応する場合を考える．

Cockburn らの考え方で習熟を表現する場合には，変更前と変更後のメニューの相違の程度により，e_i が変わると仮定することが妥当である．すなわち，あるサブメニュー内での項目の位置が変わる程度の変更への習熟は容易であるが，異なるサブメニューへと移動された変更への習熟はより難しいと想定できる．言い換えれば，項目の位置の変化に対しては，少ない試行回数で習熟し $e_i = 1$ となるが，大きな変更では $e_i = 1$ となるために必要な試行回数が増えると仮定する．このことをモデルに採り入れるためには，メニュー間の擬似距離を考える必要がある．

メニュー項目毎に（レベル，その階層でのノード番号，ノード内の位置）の 3 つ組みを考える．頻度が同じメニュー項目のノード内の位置は，メニュー項目番号でソートされているものとする．3 つ組みを (l, s, p) で表現し，l，s，p をメニュー項目順に並べたベクトル $\mathbf{L}$, $\mathbf{S}$, $\mathbf{P}$ を考える．元のメニューと新しいメニューの各々の $\mathbf{L}$, $\mathbf{S}$, $\mathbf{P}$ の組について，ハミング距離を求め，$L > S > P$ で重み付けして加えたものを，階層メニュー M_i, M_j の擬似距離 $d(M_i, M_j)$ とする．この擬似距離に応じて習熟するまでの試行回数が増えるモデル化を行う．またこの擬似距離を用いてユーザの適応コストを計算する．

4. おわりに

新たに生成されるメニューに対するユーザの適応コストまで考慮して，階層メニューを最適化するための新しい定式化を提案した．定式化した問題は，既提案の解法を若干変更することで解くことができる．

メニュー間の擬似距離と習熟速度の関係を，被験者被験者実験により確認し，具体的なモデル化を行い妥当性を検証することが今後の課題である．

参考文献

[Ahlström 05] Ahlström, D.: Modeling and improving selection in cascading pull-down menus using Fitts' law, the steering law and force fields, in *Proc. of CHI '05*, pp. 61–70 (2005)

[Amant 07] Amant, R. S., Horton, T. E., and Ritter, F. E.: Model-based evaluation of expert cell phone menu interaction, *ACM Trans. Comput.-Hum. Interact.*, Vol. 14, No. 1, pp. 1–23 (2007)

[Cockburn 07] Cockburn, A., Gutwin, C., and Greenberg, S.: A predictive model of menu performance, in *Proc. of CHI '07*, pp. 627–636 (2007)

[Gajos 08] Gajos, K. Z., Everitt, K., Tan, D. S., Czerwinski, M., and Weld, D. S.: Predictability and accuracy in adaptive user interfaces, in *Proc. of CHI '08*, pp. 1271–1274 (2008)

[松井 08] 松井 正一, 山田 誠二：遺伝的アルゴリズムによる階層メニューの最適化, 人工知能学会論文誌, Vol. 23, No. 6B, pp. 494–504 (2008)