Применение алфавитного кодирования для оптимизации интерфейса

И. В. Нечта[1,*], Б. Я. Рябко[2,3], Н. Н. Савина[2],

[1]Сибирский государственный университет телекоммуникаций и информатики, Новосибирск, Россия

[2]Институт вычислительных технологий СО РАН, Новосибирск, Россия

[3]Новосибирский государственный университет, Россия

[*]Контактный e-mail: www@inbox.ru

Рассмотрены основные подходы к оптимизации человекомашинного интерфейса и существующие алгоритмы построения иерархического меню компьютерных приложений. Предложен метод построения иерархического меню, базирующийся на психологическом законе Хика и известных в теории информации кодах Гильберта — Мура и Фано.

Ключевые слова: оптимизация человекомашинного интерфейса, закон Хика, иерархическое меню, код Гильберта — Мура, код Фано.

Введение

При создании компьютерных программ большое внимание уделяется разработке пользовательского интерфейса. Считается, что при большом обилии конкурирующих программных продуктов, которые одинаково эффективно решают задачи пользователя, предпочтение будет отдано той программе, которая имеет наиболее легкое и удобное управление, поэтому проблема поиска методов для построения удобного интерфейса привлекает внимание многих исследователей. Данная проблема относится к междисциплинарным, и ее решение может быть найдено на стыке двух наук: информатики и психологии, позволяющей изучить механизмы зрительного восприятия человека. Основные принципы восприятия изображений человеческим глазом изложены в работах [1 – 3] и в обзорной статье [4], где формулируются эргономические требования к элементам управления. Важно отметить, что в психологии восприятия известны количественные законы, используемые при планировании интерфейса. Так, скорость наведения курсора мыши и нажатия на элемент интерфейса может быть рассчитана по закону Фитса [5], устанавливающего логарифмическую зависимость между размером объекта на экране и временем наведения на него курсора. Закон Хика, впервые предложенный в статье [6], позволяет оценить среднее время осуществления выбора одного из предложенных вариантов. В соответствии с этим законом

$$t = \alpha + \beta \log w, \tag{1}$$

где t — время выбора одного из w элементов, а α, β — константы. (Здесь и ниже мы будем считать, что логарифм по основанию 2, так как переход к другому основанию влияет только на константу β.) Ряд публикаций, например [7 – 9], посвящен изучению механизмов работы мозга и особенностям реакций человека при осуществлении выбора. Считается, что закон Хика не всегда может быть применен, известны эксперименты, когда закон не находил своего подтверждения. Как было установлено позднее, результаты сильно зависят не только от предлагаемых вариантов выбора, но и от средств, которые пользователь задействует для ответной реакции (мышь, клавиатура, джойстик и т. д.). Указанные ограничения возникают вследствие особенностей алгоритмов принятия решений человеком. Тем не менее наблюдается хорошая воспроизводимость результатов при одинаковых постановках эксперимента, и большинство авторов считают, что закон Хика справедлив.

Одной из ключевых деталей интерфейса следует считать иерархическое или одноуровневое меню, где компактно размещаются имена объектов. Большое количество таких объектов вынуждает разработчиков группировать пункты меню, которое может быть представлено в виде иерархической структуры, обычно дерева. К настоящему времени известно много работ, посвященных разработке методов создания меню с минимальным средним временем доступа к каждому элементу. Среди них работы Губко и Даниленко, в которых предложен метод построения меню с близким к минимальному средним временем поиска элемента [10, 11]. Однако этот метод требует участия человека в процессе построения меню (например, на шаге 2 и 4 “Разработчик выбирает типы меню”, на шаге 5 “Разработчик разбивает множество элементов на осмысленные категории” и т. п. [10]). Во многих реальных ситуациях требуется строить меню автоматически, без участия разработчика. В частности, структура меню должна быть различной для пользователя с небольшим монитором (скажем, мобильным телефоном) и пользователя с большеэкранным компьютером. Другая ситуация, требующая автоматического построения меню, — представление информации по поисковому запросу (например, результаты заказа билетов на определенную дату), где информацию также целесообразно представлять в виде меню. Понятно, что такие меню должны строиться очень быстро, сразу после поступления запроса, и участие человека в таких случаях невозможно.

В данной работе предложен метод автоматического построения меню для случая, когда объекты могут быть упорядочены по алфавиту и пользователь знает название (имя) объекта заранее (например, при покупке авиабилета через интернет пользователь знает название аэропорта вылета и прибытия, при поиске в телефонном справочнике ему известно имя абонента и т. п.). Подчеркнем, что эта задача у́же, чем рассматриваемая в [10, 11] и ряде других работ, однако именно это позволяет найти работающий без участия человека алгоритм построения меню с временем поиска, близким к минимальному. Данный метод базируется, с одной стороны, на законе Хика, а с другой — на известных в теории информации алфавитном коде и коде Фано [12].

В статье дано описание общего алгоритма, позволяющего строить меню с временем поиска, близким к минимальному, однако трудоемкость предлагаемого метода высока, что делает актуальными задачи построения более эффективных общих методов и поиска частных подзадач, для которых могут быть найдены быстрые алгоритмы. Основное внимание уделено нахождению быстрых алгоритмов для двух частных подзадач, представляющих практический интерес, тогда как построение эффективных алгоритмов для общего случая будет задачей дальнейшего исследования.

1. Описание методов построения иерархического меню

Каждое меню характеризуется количеством уровней, а каждый из уровней — его шириной, т. е. количеством альтернатив, одна из которых должна быть выбрана.

Использование идей и методов теории информации для построения иерархических меню предлагалось в работе [13] и ряде других. В этой работе описан метод построения меню, близкого к оптимальному, однако только для случая, когда ширина, т. е. количество элементов, из которых проводится выбор, на каждом уровне меню фиксирована. В настоящей работе мы предлагаем метод построения близкого к оптимальному меню для случая, когда все объекты упорядочены в алфавитном порядке (являются словами, а не, например, картинками) и пользователь заранее знает название объекта (имя), сведения о котором ему надо найти при помощи меню.

Перейдем к более формальному описанию. Предположим, дано множество $A = \{a_1, ..., a_n\}$ и известны вероятности (частоты) обращения пользователей к его элементам $P = (p_1, ..., p_n)$.

Мы последовательно рассмотрим три представляющих практический интерес случая: 1) ширина меню на каждом уровне должна быть одинаковой, 2) ширина меню на всех уровнях заранее задана (но может быть различной на разных уровнях) и 3) наиболее общий случай — при построении меню ширина уровней не задана, а только известно, что она должна быть ограничена некоторым интервалом w^*, W^*, т. е. для ширины каждого уровня w должно выполняться неравенство $w^* \le w \le W^*$.

Заметим, что для любого меню среднее время поиска элемента (t^*) дается равенством

$$t^* = \sum_{i=1}^{n} p(a_i) \sum_{j=1}^{s(a_i)} \left(\alpha + \beta \log w_j(a_i)\right), \tag{2}$$

где $s(a_i)$ — количество уровней, или выборов, в меню, которые пользователь должен совершить для нахождения объекта a_i; $w_j(a_i)$ — ширина j-го уровня, содержащего a_i, а α и β — константы. Поясним эту формулу. Слагаемые первой суммы соответствуют элементам $a_i \in A$; $j = 1, ..., s(a_i)$ — уровни, которые пользователь проходит от верхнего уровня до a_i, причем $w_j(a_i)$ — ширина уровня j, а величина $(\alpha + \beta \log w_j(a_i))$ — время определения (выбора) перехода с уровня j на уровень $j + 1$, определяемое законом Хика (1).

Перейдем к задаче построения меню. Рассмотрим сначала первый случай. Пусть требуется построить иерархическое меню так, чтобы число элементов на каждом уровне (ширина) было равно некоторому фиксированному элементу w. Заметим, что в случае, когда все уровни одной ширины (например, w), формула (2) существенно упрощается:

$$t^* = C \sum_{i=1}^{n} p(a_i) s(a_i), \tag{3}$$

где $C = \alpha + \beta \log w(a)$ — константа. Естественный вопрос — как построить меню, для которого величина t^* в (2) (или в (3)) минимальна. В [13] эта задача решается для случая, когда все уровни меню должны быть одной ширины, т. е. рассматривается задача минимизации среднего времени (3). Оказывается, при ее решении можно использовать методы теории информации [13], точнее, методы теории кодирования источников информации, или "сжатия" данных. Дело в том, что задача построения иерархического

меню с одинаковой шириной всех уровней (т. е. с равным числом элементов на всех уровнях меню) практически идентична задаче нахождения кода с минимальной средней длиной кодового слова. Первый результат, который можно немедленно получить из близости этих двух задач, — достаточно точная нижняя оценка для t^*:

$$t^* \geq H(P), \tag{4}$$

где

$$H(P) = -\sum_{i=1}^{n} p(a_i) \log_w p(a_i) \tag{5}$$

— широко известная в теории информации энтропия Шеннона. Практическое применение этой оценки очевидно: если для какого-либо меню величина t^* близка к энтропии Шеннона (5), то меню близко к оптимальному, в противном случае стоит попытаться построить другое меню для уменьшения t^*.

В качестве примера рассмотрим задачу построения меню для множества городов $City = \{Abakan, Magadan, Moskow, Pskov, Tobolsk\}$ с заданным распределением вероятностей $P = (0.1, 0.1, 0.5, 0.25, 0.05)$, и пусть w (ширина) равна 2. По формуле (5) находим, что $H(P) = 1.88$ и из неравенства (4) получаем нижнюю оценку для t^*: $t^* \geq 1.88$.

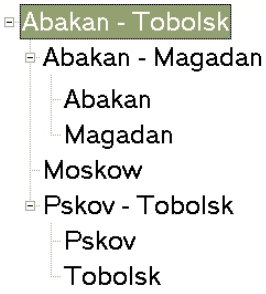
Данный частный случай рассматриваемой задачи исследован в [13], поэтому здесь мы дадим только краткое описание. Как мы отмечали, в этом случае можно использовать алгоритмы построения алфавитного кода, или кода Гильберта — Мура. Известны точный и приближенный (более простой) методы построения кода (и тем самым меню поиска), для которых справедлива оценка

$$t^* < H(P) + 2. \tag{6}$$

(Для оптимального меню t^* может быть несколько меньше, чем для приближенного. Однако для обоих алгоритмов справедливо данное неравенство.) Сравнивая (6) с нижней оценкой (4), мы видим, что построенное обоими алгоритмами меню довольно близко к оптимальному. Мы не описываем указанные алгоритмы, а даем описание модифицированного метода, сочетающего свойства указанных алгоритмов и кода Фано (иногда называемого кодом Шеннона — Фано) [12]. Кроме того, указанный метод естественным образом обобщается на второй случай (задача построения меню с заданными уровнями разной ширины), и мы дадим его описание именно для этого случая, так как изменения метода для первого случая, когда все уровни одной ширины (w), очевидны.

Итак, пусть дано множество $A = \{a_1, ..., a_n\}$, известны вероятности (частоты) обращения пользователей к его элементам $P = (p_1, ..., p_n)$ и требуется построить меню с уровнями разной ширины $W = \{w_1, ..., w_r\}$. Описание всех этапов построения меню по предлагаемому методу будем иллюстрировать на вышеупомянутом примере $A = City$, $P = (0.1, 0.1, 0.5, 0.25, 0.05)$, $W = \{3, 2, 3, 2\}$.

На первом шаге все множество объектов разбивается на w_1 с сохранением алфавитного порядка (так, что каждое подмножество является частью исходного списка) и суммы вероятностей элементов подмножеств близки друг к другу (и к $1/w_1$). В нашем примере $w_1 = 3$ и разбиение проводится следующим образом: $(Abakan, Magadan)$, $(Moskow)$, $(Pskov, Tobolsk)$, соответствующие суммы вероятностей подмножеств 0.2, 0.5, 0.3. Затем каждое из вновь полученных подмножеств разбивается на w_2 подмножеств так, что суммы вероятностей объектов в подмножествах близки друг к другу.



Построенное меню

Эти подмножества задают второй уровень иерархии меню. Затем каждое из вновь полученных подмножеств разбивается на w_3 подмножеств и т. д. В нашем примере первое и третье подмножества разбиваются на два и процесс завершается. Полученное меню показано на рисунке.

Для сравнения применим этот метод для построения меню при фиксированной ширине одного уровня $w = 2$. В этом случае при работе алгоритма сначала все множество разбивается на две части: $(Abakan, Magadan, Moskow)$, $(Pskov, Tobolsk)$. На втором шаге первое подмножество разбивается на два: $(Abakan, Magadan)$, $(Moskow)$, а второе — на $(Pskov)$ и $(Tobolsk)$. Наконец, на последнем шаге подмножество $(Abakan, Magadan)$ разбивается на два и работа над меню заканчивается. По формуле (2) находим, что для этого меню $t^* = 2.2$, это довольно близко к значению найденной ранее нижней оценки 1.88.

Описанный метод довольно прост, причем легко подсчитать, что время работы описанного алгоритма при больших n не превосходит $O(n^2)$.

Рассмотрим теперь общий (третий) случай: при построении меню ширина уровней не задана, а только известно, что она должна быть ограничена некоторым интервалом w^*, W^*, т. е. для ширины каждого уровня w должно выполняться неравенство

$$w^* \le w \le W^*. \tag{7}$$

В этом случае оптимальным будет меню, для которого среднее время поиска, задаваемое (2), минимально (при выполнении (7)). Здесь, по аналогии с рассмотренными случаями, также можно использовать приближеный метод, описание которого мы разобьем на шаги. На первом шаге множество A разбиваем на подмножества A_1, $A_2, ..., A_w$ с сохранением заданного на A порядка (т. е. $a < b$ для всех $a \in A_i$, $b \in A_j$, если $i < j$) и выполнением неравенства (7). (Здесь $|X|$ — число элементов в множестве X.) Для дальнейшего описания определим

$$k_0 = 0, \quad k_i = \sum_{j=1}^{i} |A_j|, \quad p(A_i) = \sum_{j=k_{i-1}}^{k_i} p(a_j), \quad i = 1, ..., w.$$

Главное условие, которому должно удовлетворять данное разбиение, — величина

$$(\alpha + \beta \log w) + \sum_{i=1}^{w} p(A_i) \sum_{j=k_{i-1}}^{k_i} (\alpha + \beta \log |A_i|)$$

минимальна (среди всех разбиений). Поясним эту формулу. Величина $(\alpha + \beta \log w)$ — это время, затрачиваемое на выбор на первом уровне, задаваемая законом Хика (1),

величины $(\alpha + \beta \log |A_i|)$ — оценки времени, затрачиваемого на выбор на следующих уровнях. На втором шаге указанная процедура повторяется с каждым из подмножеств A_1, A_2 ,..., A_w, затем с каждым из вновь полученных подмножеств и т. д., до тех пор, пока в каждом из полученных подмножеств не останется по одному элементу.

При очевидной непосредственной реализации этого метода его время работы как функция числа объектов n растет экспоненциально. При использовании известного метода "ветвей и границ" [14] сложность может быть понижена до полиномиальной, однако она довольно высока. Построение более эффективного алгоритма остается задачей дальнейшего исследования.

Заключение

Задача построения деревьев поиска и им подобных структур встречается в теории информации, теории сложности алгоритмов и других разделах дискретной математики. Оптимальные и близкие к оптимальным кодовые деревья (они же поисковые деревья) были получены во второй половине прошлого века для многих случаев и описаны в многочисленных монографиях еще в 1960- и 1970-е годы, причем были найдены ставшие классическими подходы к оценке сложности алгоритмов. В те же годы было показано, что энтропия Шеннона является достаточно точной нижней границей для средней длины кода (и среднего времени поиска в дереве). Построение иерархического меню фактически является задачей нахождения дерева поиска, поэтому вполне естественно использовать подходы, идеи и методы теории информации и теории сложности алгоритмов, где задачи построения оптимальных деревьев поиска рассматривались в разных постановках и успешно решались. В данной работе рассмотрена задача построения иерархического меню для случая, когда объекты упорядочены и для нее предложен алгоритм, который может работать без участия человека. При этом данный метод позволяет использовать закон Хика для минимизации времени поиска элементов в меню.

Работа может рассматриваться как первый шаг в направлении разработки эффективных методов построения близких к оптимальным иерархических меню, так как в этой области многие задачи далеки от полного решения. Среди новых направлений исследований стоит отметить разработку метода автоматического построения меню для общего случая, а также учет не только закона Хика, но и других законов психологии восприятия при построении меню.

Благодарности. Работа выполнена при финансовой поддержке РФФИ (грант № 15-07-01851).

Список литературы / References

[1] **Allport, F.H.** Theories of perception and the concept of structure. N.Y.: John Wiley and Sons, 1955. 709 p.

[2] **Gibson, J.J.** The ecological approach to visual perception. Boston: Houghton Miffin, 1986. 332 p.

[3] **Савина Н.Н.** Психология зрительного восприятия программных средств образовательного назначения. Новосибирск: Сибирское соглашение, 2003. 20 с.
Savina, N.N. The psychology of visual perception of educational software applications. Novosibirsk: Sibirskoe Soglashenie, 2003. 20 p. (in Russ.)

[4] **Рыжов В.А., Корниенко А.В., Демидович Д.В.** Качество экранных изображений в обучающих программах // Педагогическая информатика. 2002. Т. 1. С. 42–55.
Ryzhov, V.A., Kornienko, A.V., Demidovich, D.V. Quality of the screen image in training programs // Pedagogical Informatics. 2002. Vol. 1. P. 42–55. (in Russ.)

[5] **Fitts, P.M.** The information capacity of the human motor system in controlling the amplitude of movement // Journal of Experimental Psychology. 1954. Vol. 47(6). P. 381–391.

[6] **Hick, W.E.** On the rate of gain of information // Quarterly Journal of Experimental Psychology. 1952. No. 4. P. 11–26.

[7] **Dassonville, P. et al.** Choice and stimulus–response compatibility affect duration of response selection // Cognitive Brain Research. 1999. No. 7(3). P. 235–240.

[8] **Wright, C.E., Marino, V.F., Belovsky, S.A., Chubb, C.** Visually guided, aimed movements can be unaffected by stimulus-response uncertainty // Experimental Brain Research. 2007. No. 179. P. 475–496.

[9] **Leonard, J.A.** Tactual choice reactions. I // Quarterly Journal of Experimental Psychology. 1959. No. 11. P. 76–83.

[10] **Губко М.В., Даниленко А.И.** Математическая модель оптимизации структуры иерархического меню // Проблемы управления. 2010. № 4. С. 49–58.
Gubko, M.V., Danilenko, A.I. Theory for hierarchical menu structure optimization // Problemy Upravleniya. 2010. No. 4. P. 49–58. (in Russ.)

[11] **Губко М.В., Даниленко А.И.** Оптимизация пользовательских меню с учетом семантического качества // Проблемы управления. 2012. № 2. С. 53—63.
Gubko, M.V., Danilenko, A.I. Semantic-aware optimization of user interface menus // Problemy Upravleniya. 2012. No. 2. P. 53–63. (in Russ.)

[12] **Cover, T.M., Thomas, J.A.** Elements of information theory. N.Y.: Wiley-Interscience, 2006. 776 p.

[13] **Witten, I.H., Cleary, J.G., Greenberg, S.** On frequency-based menu-splitting algorithms // Intern. Journal of Man-Machine Studies. 1984. Vol. 21, No. 2. P. 135–148.

[14] **Ахо А.В., Хопкрофт Д., Ульман Д.** Структуры данных и алгоритмы. М.: Изд. дом "Вильямс", 2003. 384 с.
Aho, A., Ullman, J., Hopcroft, J. Data structures and algorithms. Amsterdam: Addison-Wesley, 1983. 436 p.

Поступила в редакцию 8 июня 2015 г.,
с доработки — 6 августа 2015 г.

Alphabetical coding for interface optimization

Nechta, Ivan V.[1,*], Ryabko, Boris Ya.[2,3], Savina, Nadezhda N.[2]

[1]Siberian State University of Telecommunication and Informatics, Novosibirsk, 630102, Russia
[2]Institute of Computational Technologies SB RAS, Novosibirsk, 630090, Russia
[3]Novosibirsk State University, Novosibirsk, 630090, Russia

*Corresponding author: Nechta, Ivan V., e-mail: www@inbox.ru

Basic approaches for optimization of a human-machine interface were discussed. The existing algorithms for constructing a hierarchical menu for computer applications are briefly discussed.

Consider the class of problems for which the objects can be arranged in alphabetical order and a user knows the name of the object in advance. For this class of problems a menu could be arranged automatically (i. e. not requiring human participation). Such method of construction allows minimization of the average search time.

The proposed method of constructing a hierarchical menu is based on a psychological Hick's law and known in the information theory as Gilbert-Moore and Fano codes. It is shown that the Shannon entropy is the lower limit of the average search time for any menu.

The article describes a general algorithm, that allows to build a menu whose search time is close to the minimum, but the complexity of the proposed method is high. It makes the discussed problems (first of all, designing more effective general methods for the search and, secondly, for the sub-tasks search, for which fast algorithms could be found) to be very actual.

The considered examples have shown that the proposed method allows to build the menu which is close to optimal.

Keywords: optimized man-machine interface, Hick's Law, a hierarchical menu, the Gilbert-Moore code, the Fano code.

Acknowledgements. This work was financially supported by RFBR (grant No. 15-07-01851).

Received 8 June 2015
Received in revised form 6 August 2015