УДК 004.512.3(45)

Алгоритм построения алфавитного меню полиномиальной сложности

И. В. Нечта[1]

В настоящей работе предложен алгоритм автоматического построения алфавитного меню, имеющий квадратичную трудоемкость. Указанный алгоритм позволяет использовать ограничения, накладываемые на ширину уровней меню. В работе также представлены результаты экспериментальных исследований времени работы пользователя с меню до оптимизации и после. Показано, что оптимизация становится эффективной при размере меню 700 и более элементов.

Ключевые слова: оптимизация меню, закон Хика, оптимизация интерфейса.

1. Введение

Одним из важных этапов создания программного обеспечения является разработка пользовательского интерфейса. Известно, что пользователь в конечном итоге сравнивает программные продукты по качеству интерфейса. В этой связи большое количество научных работ посвящено оптимизации скорости человеко-машинного взаимодействия. Считается, что решение данной проблемы лежит на стыке нескольких наук: психологии, нейрофизиологии и др. Однако в ряде работ, например, в [1, 2], предлагалось использовать методы теории информации и математической статистики. В действительности работа человеческого мозга при взаимодействии с интерфейсом может быть описана следующими закономерностями. Закон Хика, предложенный в работе [3], являющийся, по сути, аналогом энтропии Шеннона, устанавливает логарифмическую зависимость между временем, за которое пользователь осуществляет выбор между элементами интерфейса, и их количеством:

$$t = \alpha + \beta * \log_2 N, \quad (1)$$

где t – время выбора, α и β – константы, соответствующие навыкам пользователя, N – количество элементов, из которых осуществляется выбор. Другие закономерности, например, закон Фитса [4], описывают время наведения и нажатия курсора мыши по элементам интерфейса:

$$t = \alpha + \beta * \log_2 \frac{D}{W}, \quad (2)$$

где α и β – константы, D – дистанция перемещения курсора и W – ширина цели.

Одним из важных элементов интерфейса является меню. Выбор оптимальной иерархии меню обеспечивает высокую скорость работы пользователя с программой. Ряд работ посвящен созданию методов построения меню близких к оптимальным, например, в статье [5], однако требуется участие человека при разбиении элементов меню на осмысленные категории. В настоящей статье автором предлагается метод автоматического построения алфавитного меню, когда имеются ограничения на ширину одного уровня.

[1]Работа выполнена в рамках государственного задания рег. № 4А-А16-116070610039-4 от 6.07.2016 г.

2. Описание алгоритма

В предыдущих работах [6, 7] автором рассматривались три случая, представляющие практический интерес.

Первый случай, когда ширина меню всегда одинакова. В такой ситуации было предложено использовать алгоритм автоматического построения меню, базирующийся на алфавитном коде Гилберта–Мура.

Второй случай, когда ширина меню заранее задана для всех уровней и может различаться на разных уровнях. Предложенный в работе [6] алгоритм, по сути, является модификацией предыдущего алфавитного кода, в котором используются различные основания систем счисления, соответствующие ширинам уровней меню. Полученные таким модифицированным алгоритмом кодовые слова однозначно определяют позицию любого элемента в иерархии меню. Также алгоритм позволяет соблюдать заданные ограничения ширины уровня.

Третий, наиболее общий случай, когда ширина меню не задана, а лишь известно, что она должна быть ограничена некоторым интервалом, т.е. выполняется неравенство

$$a \le w_j \le b, \tag{3}$$

где a и b – начало и конец интервала, а w_j – ширина[2] j-го уровня меню. Такая задача часто возникает при проектировании интерфейса мобильных приложений, где существенны ограничения на размер экрана, или когда учитываются психологические особенности зрительного восприятия человека. Среднее время поиска элемента в таком меню вычисляется по формуле:

$$t = \sum_{i=1}^{n} p(x_i) \sum_{j=1}^{s(x_i)} \alpha + \beta * \log_2[w_j(x_i)], \tag{4}$$

где x_i – пункт меню; $p(x_i)$ – вероятность выбора x_i-го пункта меню; $s(x_i)$ – количество уровней меню, совершаемых пользователем при выборе элемента x_i; α и β – коэффициенты, определяемые законом Хика; $w_j(x_i)$ – ширина j-го уровня меню, содержащего элемент x_i.

В соответствии с формулой (4) для расчета среднего времени, которое затрачивается на выбор в текущем уровне меню, мы вынуждены рекурсивно вычислять аналогичное время для каждого нижележащего уровня. Более того, полный перебор всех вариантов ширины w_j меню от a до b на каждом уровне делает алгоритм экспоненциально сложным. При среднестатистическом размере меню порядка 300 элементов время оптимизации меню станет неприемлемо большим. В настоящей работе предложен быстрый алгоритм, имеющий трудоемкость $O(n) \approx n^2$.

Теперь опишем предлагаемый алгоритм. Предварительно введем меру M_j для оценки качества группировки, эквивалентную упрощенной оценке среднего времени поиска элемента, представленной в формуле (1). Данная мера вычисляется по формуле (4) для текущего j-го уровня меню с допущением, что каждый последующий уровень, $(j+1)$-й, содержит все элементы в несгруппированном виде, т.е. уровня $(j+2)$ не существует. Такое упрощение позволяет избавиться от рекурсивного вызова расчета оценки для всех подуровней. На этапе расчета меры мы можем пренебречь ограничением формулы (3), накладываемым на ширину $(j+1)$-го уровня.

Теперь рассмотрим алгоритм построения оптимизированного алфавитного меню. Пусть в меню имеется N элементов. Получим разбиение для самого верхнего (текущего) уровня.

Алгоритм 1. Алгоритм разбиения текущего уровня на подуровни.

Шаг 1. Предположим, что наименьшее (т.е. наиболее выгодное) значение M_j достигается при $w_j = a$.

[2] Здесь под шириной понимается количество пунктов меню (в т.ч. группирующих), которые будут видны пользователю на данном уровне.

Шаг 2. Если в текущем уровне j содержится более чем w_j элементов, то объединим (так называемое «слияние») два элемента данного уровня в группу. Слияние производится только над двумя элементами[3], следующими друг за другом. Указанные элементы выбираются так, чтобы сумма их вероятностей была минимальна среди других аналогичных элементов. Выполняем шаг 2 до тех пор, пока ширина уровня не достигнет значения w_j.

Шаг 3. Вычислим меру M_j для текущего уровня меню.

Шаг 4. Увеличим значение w_j на единицу и, если выполняется неравенство $w_j \leq b$, то возвращаем исходное состояние уровня меню (до слияний) и переходим к шагу 2.

Шаг 5. Строим итоговый уровень меню с шириной, для которой достигается минимум значения M_j.

Аналогично производится разбиение всех остальных подуровней. Таким образом, с одной стороны, мы избавляемся от крайне трудоемкой рекурсии при вычислении меры M_j, и с другой стороны, выполняем ограничения, накладываемые на ширину уровней.

3. Результаты экспериментального исследования

3.1. Описание проводимого эксперимента

Теперь проведем экспериментальное исследование эффективности нашего алгоритма. Измерим среднее время поиска элемента для двух случаев:

- одноуровневое меню (без оптимизации);
- многоуровневое меню (с оптимизацией).

Для работы нашего алгоритма зададим начальные параметры. Будем использовать закон Ципфа для распределения вероятностей выбора элементов меню, как это было сделано в работе [5]. Для расчета меры качества уровня меню возьмем значения $\alpha = 50$ мсек и $\beta = 150$ мсек. Границы ширины уровня меню положим следующие: $2 \leq w_j \leq 8$.

Эксперимент поставлен следующим образом. На экране мобильного телефона возникает всплывающая подсказка, в которой указывается задание (наименование города, который пользователь должен найти). Задание выдается с учетом рассмотренного ранее распределения вероятностей, соответствующего закону Ципфа. Сразу после получения задания испытуемый находит соответствующий пункт меню (либо пользуясь элементом прокрутки, либо переходит по иерархии меню до достижения искомого элемента) и нажимает по нему. В эксперименте замеряется время выдачи задания и время правильного нахождения элемента. Если пользователь неправильно выбрал элемент, то такой результат отбрасывался.

В эксперименте принимало участие 20 студентов. Одна половина тестируемых работала с неоптимизированным меню, другая с оптимизированным. Одному студенту для каждого размера меню генерировалось по 30 заданий. В итоге были получены результаты, представленные в табл. 1, 2.

Отметим, что в ходе работы алгоритма ширина уровня меню всегда оказывалась максимальной, т.е. всегда $w_j = b$. Этот факт является вполне ожидаемым, т.к. анализируя закон Хика, мы видим аналогичную закономерность: чем шире уровень, тем быстрее поиск. В дальнейшем планируется добавить в меру M другие закономерности (например, закон Фитса и др.), что существенно повлияет на итоговое значение w_j.

Исходя из анализа графика, представленного на рис. 1, мы можем сделать вывод, что оптимизация становится эффективна при размере меню 700 и более элементов.

[3] В качестве элемента может выступать конечный или группирующий пункт меню.

Таблица 1. Среднее время поиска элемента по неоптимизированному меню

Общее количество элементов в меню	Среднее время поиска, сек
100	07.13
300	11.97
500	13.35
700	13.04
900	13.82
1100	15.70

Таблица 2. Среднее время поиска элемента по оптимизированному меню

Общее количество элементов в меню	Среднее время поиска, сек
100	11.24
300	14.81
500	14.39
700	13.02
900	12.68
1100	13.68

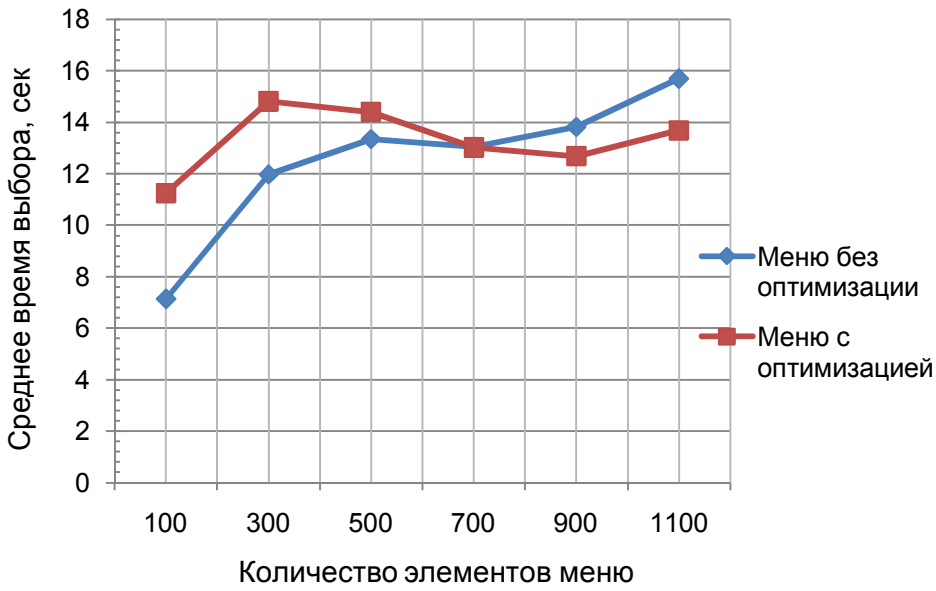

Рис. 1. График зависимости среднего времени поиска элемента от размера меню

3.2. Оценка трудоемкости полученного алгоритма

Теперь оценим скорость работы нашего алгоритма. Проведем расчет его трудоемкости. Для этого введем дополнительные обозначения:

$[a; b]$ – границы ширины меню, N – количество пунктов во всем меню; n_j – количество элементов меню j-го уровня.

Таким образом, получим: количество подбираемых вариантов ширины произвольного уровня меню w_j равно $b - a + 1$; количество слияний на первом уровне составит $N - b$; трудоемкость одного слияния: $b + \frac{N-b}{2} = \frac{N+b}{2}$; трудоемкость работы алгоритма на верхнем уровне составит:

$$O(n) = (b - a + 1)(N - b)(\frac{N+b}{2}). \quad (5)$$

Количество уровней во всем меню, которые будут иметь группировки[4] (на которых будет выполняться процесс слияния), равно $\log_b N - 1$. Количество меню j-х уровней составляет b^{j-1}; количество элементов на j-м уровне меню составит: $\frac{N}{b^{j-1}}$. Таким образом, получится общая трудоемкость алгоритма, представленная ниже:

$$O(n) = \sum_{j=1}^{\log_b(N)-1} \left[(b - a + 1)(b^{j-1})(\frac{N}{b^{j-1}} - b)(\frac{\frac{N}{b^{j-1}}+b}{2})\right] =$$

$$O(n) = (b - a + 1) \sum_{j=1}^{\log_b(N)-1} \left[(N - b^j)(\frac{N}{2b^{j-1}} - \frac{b}{2})\right] = \quad (6)$$

$$O(n) = (b - a + 1) \sum_{j=1}^{\log_b(N)-1} \left[(N - b^j)(\frac{N-b^j}{2b^{j-1}})\right].$$

В итоге у нас получился алгоритм, который имеет квадратичную трудоемкость и позволяет задавать ширину уровня меню диапазоном значений. В дальнейшем планируется дополнить меру M другими закономерностями и рассмотреть случай, когда пользователь может отсеивать часть данных с помощью предварительного интерактивного поиска.

Литература

1. *Leonard J. A.* Tactual choice reactions: I // Quarterly Journal of Experimental Psychology. 1959. V. 11, № 2. P. 76–83.
2. *Dassonville P., Lewis S. M., Foster H. E., Ashe J.* Choice and stimulus–response compatibility affect duration of response selection // Cognitive Brain Research. 1999. V. 7, № 3. P. 235–240.
3. *Hick W. E.* On the rate of gain of information // Quarterly Journal of Experimental Psychology. 1952. V. 4. P. 11–26.
4. *Fitts P. M.* The information capacity of the human motor system in controlling the amplitude of movement // Journal of Experimental Psychology. 1954. V. 47 (6). P. 381–391.
5. *Witten I. H., Cleary J. G., Greenberg S.* On frequency-based menu-splitting algorithms // International Journal of Man-Machine Studies. 1984. V. 21, № 2. P. 135–148.
6. *Нечта И. В., Рябко Б. Я., Савина Н. Н.* Применение алфавитного кодирования для оптимизации интерфейса // Computational technologies. 2015. Т. 20, № 5. С. 97–104.
7. *Нечта И. В.* Построение меню при помощи алфавитного кода // Вестник СибГУТИ. 2015. № 4. С. 40–46.

Статья поступила в редакцию 17.10.2016

Нечта Иван Васильевич
к.т.н., доцент кафедры прикладной математики и кибернетики, начальник отдела подготовки кадров высшей квалификации СибГУТИ (630102, Новосибирск, ул. Кирова, 86),
тел. (383) 269-83-59, e-mail: tal200@lenta.ru.

[4] Здесь мы рассматриваем самый простой случай, когда выбор элементов меню равновероятен.

Alphabetical menu construction algorithm of polynomial complexity

I. Nechta

In this paper, we propose an algorithm for automatic construction of alphabetical menu with polynomial-quadratic complexity. This algorithm allows us to use restrictions on the width of the menu levels. The results of experimental research of user working time with the menu before and after optimization are presented. It is shown that our optimization becomes effective when the size of the menu items equals 700 or larger.

Keywords: menu optimization, Hick`s law, interface optimization.