УДК 004.9

сРАВНЕНИЕ ОЦЕНОК СЛОЖНОСТИ ХОЛСТЕДА И КОЛМОГОРОВА

**Цветков В.Я., Терентьев П.В.**

Федеральное государственное бюджетное образовательное учреждение высшего образования "МИРЭА - Российский технологический университет", 119454, Россия, г. Москва, проспект Вернадского, 78, e-mail: cvj2@mail.ru

Исследуется относительно новый метод оценки сложности программ по Холстеду. Описан метод оценки вычислительной сложности по Колмогорову. Показана близость идей оценки. В одном случае (Колмогоров) оценивается обрабатываемая последовательность и программная последовательность. В другом случае (Холстед) оценивается программная последовательность и последовательность словаря или алфавита. Показана целесообразность введения термина программная конструкция, чтобы отличать описание программы от ее реализации.

Ключевые слова: программа, стандартизация, сложность, Колмогоровская сложность, сложность по Холстеду, программная конструкция.

COMPARISON OF ESTIMATES OF COMPLEXITY OF HOLSTEAD AND KOLMOGOROV

**Ttsvetkov V. Ya., Terentyev P.V.**

*Federal State Educational Institution of Higher Education “Russian Technological University” (RTU MIREA), 119454, 78 Vernadsky Avenue, Moscow, Russia e-mail:cvj2@mail.ru*

**The article explores a relatively new method for estimating the complexity of the Halstead programs. A method for estimating computational complexity according to Kolmogorov is described. The article shows the closeness of ideas for assessing complexity. In one case (Kolmogorov), the processed sequence and the program sequence are evaluated. In another case (Halstead), the program sequence and the sequence of a dictionary or alphabet are evaluated. The article recommends the introduction of the term “software construct” to distinguish the description of the program from its implementation.**

Key words: program, standardization, complexity, Kolmogorov complexity, Halstead complexity, software construction.

**Колмогоровская сложность как основа работ Холстеда**

Идея метрик Холстеда заложена в работах Колмогорова по исследованию вычислительной сложности. Исходя из сравнительно невысокого уровня (в сравнении с современными исследованиями) аналитического обеспечения программ того времени, Колмогоров использовал для оценки сложности идеи комбинаторики. В основе понятия Колмогоровской сложности [1] лежит понятие декомпрессора – произвольной вычислимой функция из N в N, что является ее формальным определением. Неформальное определение Колмогоровской сложности иллюстрируется таким примером. Нужно послать большой объем информации. Для этого ее необходимо положить ее в некий компрессор. Сжать информацию и отправить сжатое описание информации. Сжатая информация является примером Колмогоровской сложности, а декомпрессор – разжимающий ее механизм – произвольной вычислимой функцией. Здесь напрашивается параллель с теоремой отсчетов Хартли - Найквиста-Шеннона – Котельникова. Сжатие информации имеет ограничение по семантике, а не по объему. Сжимать можно только так. что бы можно было бы разжать без потери содержательности информации. Отсюда Колмогоров определяет сложность последовательности как длину самого короткого сообщения, при котором декомпрессор восстанавливает исходную информацию (х). ПО существу речь иде о принципе минимальной длины линейного сообщения. То есть для нелинейных и многомерных сигналов данный подход не приемлем. Далее Колмогоров делает еще одно упрощение. Он рассматривает код из нулей и единиц. Тогда, согласно принципу сообщения минимальной длины правильнее будет та программа, которая содержит меньше всего символов. Эта условность требует использования одной системы счисления – только двоичной. Число 15 в шестнацатиричной системе счисления потребует один символ, в десятичной системе счисления – два символа, а в двоичной – четыре символа. Самой кроткой будет последовательность в шестнацатиричной системе. Поэтому сложность Колмогорова связана с простой системой. Сложность Колмогорова зависит от отношения длинны программы обрабатывающей последовательность к самой длине последовательности.

 Последовательность считается сложной, когда ее невозможно вывести программой, чья длина меньше, чем длина последовательности плюс ненулевая константа, зависящая от ЭВМ. Последовательность считается сложной, когда не существует закономерности кодирования или она не найдена. Это очень важное замечание, которое показывает условность понятия сложность. Последовательность считается простой, когда ее можно вывести программой, чья длина меньше, чем длина последовательности плюс ненулевая константа, зависящая от ЭВМ. Последовательность считается простой, когда найдена и существует закономерность кодирования. Эти определения являются стандартизованными [2] и объективными.

Любовь математиков к единицам и нулям обусловлена тягой к математической логике и надеждой на работу закона «исключения третьего». Если переходить от последовательности как некой внешней субстанции к последовательности, которая определяет саму программу, то перейдем к метрикам Холстеда. Колмогоров рассматривал последовательность как объект обработки и сравнивал с последовательностью программы. Холстед перенес рассмотрение только на последовательность программы и анализировал ее структуру.

**Подход Холстеда**

Особенность подхода Холстеда в том, что он рассматривает программу двояко как информационную конструкцию [3] и как ее реализацию. Это дает основание ввести термин программная конструкция для различения программы на языке от программы в компьютере. Метрики Холстеда вычисляются на основании анализа синтаксических элементов исходного кода программы, введенные Морисом Ховардом Холстедом в 1977 году [4, 5].

Колмогоров рассматривает две последовательности: последовательность обработки и последовательность – программу. Холстед рассматривает также две последовательности: последовательность программу и последовательность – словарь. Соответственно длина программы N и длина словаря η. Словарь можно рассматривать как алфавит некого языка или как набор связанных информационных единиц [6].

Для длины строки $N,$ которая состоит из символов словаря η, определяются некоторые требования. Первым требование выходит из того что каждый символ словаря должен присутствовать в строке хотя бы один раз, исходя из этого получим неравенство:

η ≤ $N$

Это неравенство определяет нижнюю границу для длины строки $N$, выраженную через символы входящие в словарь η. Для определения верхней границы разделим длину строки $N$ на подстроки длины η.

$N$/ η

После разделения нашей программы получается, что она состоит из $N$/ η операторов длины η каждый. Для дальнейшего поиска длины нужно исключить дубликаты подстрок длины η, после чего появится искомая верхняя граница длины строки. Требование отсутствие дубликатов подстрок длины η является важным моментом для программ, так как из экономии выражений получается, что общему подвыражению присваивается отдельное им, поэтому его нужно вычислять всего один раз. Следовательно, если в случае программе потребуется использовать общее подвыражение длины η более 1-го, присваивание его отдельному операнду увеличится η на одну единицу. Методом индукции можно найти число комбинаций из η элементов, взятых по η за один раз.

Следовательно, программа будет состоять не более чем из ηη подстрок длины η, тогда верхняя граница будет следующей:

$N$ ≤ ηη1+1

Попробуем уточнить верхнюю границу предыдущего неравенства. Исходя, из того, что словарь η состоит из операторов (η1) и операндов (η2), которые, как правило, чередуются. При η=4 и двух операторов, и двух операндов, наблюдаем, что при 44=256, возможны лишь некоторые комбинации

После замены преобразуем уравнение:

$N$ ≤ η \* η1η1 \* η2η2

В эту верхнюю границу должно входить не только упорядоченное множество, состоящее из N элементов, которое представляет исследуемую программу, но и все его подмножества. Семейство всех подмножеств нашего множества из N элементов включает 2N элемента, и оно называется множеством-степенью. Теперь мы можем приравнять наше равнение к числу элементов в множестве-степени и из этого выявить длину алгоритма, через его словарь:

2N = η1η1 \* η2η2

Логарифмируем выражение

$N $= log2(η1η1 \* η2η2)

По свойству логарифмов преобразуем

$N$ = log2η1η1 + log2η2η2

Из этого получим

$\hat{N}$ = log2η1η1 + log2η 2η2

$\hat{N}$ =η1log2η1 + η2log2η2

Чтобы различать длину программы N от вычисленной длины для N сверху добавим специальный знак « $\hat{ }$ ».

Соотношение между расчетной длиной программной конструкции Ň и фактической характеризует сложность по Холстеду. Метрическая характеристика, которая соответствует размеру алгоритмической реализации программной конструкции, называется объёмом $V$и определяется уравнением

$V$ = $N$ \* log2η

Где $N$ – это длина реализации ($N$1 + $N$2); η – её словарь (η1 + η2). Данная интерпретация отображает объем программы в битах.

Понятно, что перевести алгоритм с одного языка на другой, его объем изменится. На примере перевода алгоритма с Фортрана в машинный код его объем увеличится. А если написать алгоритм на, каком-нибудь развитом языке, нежели тот, на котором он был написан, и тогда в этом случае его объем уменьшится.

Если перевести алгоритм с одного языка на другой, его объем изменится. Перевода алгоритма с Фортрана в машинный код его объем увеличится. Написание алгоритма на более развитом языке, по сравнению с тем, на котором он был написан, приведет к уменьшению объема. Эти рассуждения касались выбора кода выше при анализе сложности Колмогорова. Если программная конструкция реализована, то объем ее изменится. Звездочками будем обозначать сжатые формы параметров алгоритма, следовательно, из уравнения $V$= $N$\* log2η получим минимальный (потенциальный) объем.

$V$\* = ($N$\*1 + $N$\*2)log2(η\*1 + η\*2)

В минимальной форме операторы и операнды не нуждаются в повторениях, поэтому получаем

$N$\*1= η\*1 и $N$\*2= η\*2

Тогда изменяем формула потенциального объема

$V$\* = (η\*1 + η\*2 )log2(η\*1 + η\*2)

Каждый алгоритм должен включать символ присваивания, группировки или один оператор для имени функции, это и будет минимальным возможным числом операторов η\*1 = 2. Преобразуем уравнение потенциального объема

$V$\* = (2+ η\*2 )log2(2+ η\*2)

Где η\*2 представляет собой число различных входных, а также выходных параметров.

Потенциальный объем $V$\* любого алгоритма не должен быть зависимым от языка программирования, на котором он будет выражен. Если η\*2 рассматривается, как число единых по смыслу рассматриваемых операндов, значит, $V$\* является наиболее полезной мерой содержания алгоритма.

Остановимся на данном моменте чтобы подчеркнуть сходство идей Колмогорова и Холсдеда в оценке сложности линейных конструкций, но в одном случае обработка и процессуальная модель. в другом случае дескриптивная модель [7].

Список литературы

1. Верещагин Н.К., Успенский В.А., Шень А. Колмогоровская сложность и алгоритмическая случайность: М.: МЦНМО, 2013 - 576с

2. Цветков В.Я. Стандартизация информационных программных средств и программных продуктов. - М.: МГУГиК, 2000 - 116с.

3 Tsvetkov V. Ya. Information Constructions // European Journal of Technology and Design. -2014, Vol (5), № 3. - p.147-152.

4. Холстед М. Х. Начала науки о программах. М., Финансы и статистика. 1981, 128 стр.

5. Цветков, В.Я., Буравцев В.А. Метрики сложной детерминированной системы // Онтология проектирования. – 2017. – Т. 7, №3(25). - С. 334-346

6. V. Yа. Tsvetkov.Semantic environment of information units // European researcher. Series A. 2014, Vol.(76), № 6-1, p. 1059-1065.

7. Раев В.К. Процессуальные и дескриптивные информационные модели // Славянский форум. -2018. – 3(21). - с.28-32.

 References

1. Vereshchagin N.K., Uspenskij V.A., SHen' A. Kolmogorovskaya slozhnost' i algoritmicheskaya sluchajnost': M.: MCNMO, 2013 - 576s

2. Cvetkov V.YA. Standartizaciya informacionnyh programmnyh sredstv i programmnyh produktov. - M.: MGUGiK, 2000 - 116s.

3 Tsvetkov V. Ya. Information Constructions // European Journal of Technology and Design. -2014, Vol (5), № 3. - p.147-152.

4. Holsted M. H. Nachala nauki o programmah. M..: Finansy i statistika. 1981. - 128 s.

5. Cvetkov, V.YA., Buravcev V.A. Metriki slozhnoj determinirovannoj sistemy // Ontologiya proektirovaniya. – 2017. – T. 7, №3(25). - S. 334-346.

6. V. Yа. Tsvetkov.Semantic environment of information units // European researcher. Series A. 2014, Vol.(76), № 6-1, p. 1059-1065.

7. Raev V.K. Processual'nye i deskriptivnye informacionnye modeli // Slavyanskij forum. -2018. – 3(21). - s.28-32.