УДК 510.52

Малов Д. Н., канд. техн. наук, доц.,
Национальный исследовательский университет – Высшая школа экономики

Ульянов М. В., д-р техн. наук, проф.,
Московский государственный университет печати,
Национальный исследовательский университет – Высшая школа экономики

Метод получения функции трудоёмкости рекурсивных алгоритмов на основе анализа порожденного дерева рекурсии

Аннотация

В статье излагается метод анализа ресурсной эффективности рекурсивных алгоритмов в аспекте трудоёмкости, основанный на исследовании порожденного дерева рекурсии. Метод позволяет получить функцию трудоемкости для рекурсивных алгоритмов в модели вычислений процедурного языка высокого уровня. Дополнительной особенностью предлагаемого метода является возможность получения функции объема памяти исследуемого алгоритма.

Введение.

Многие актуальные задачи экономико-математического моделирования могут быть эффективно решены с помощью рекурсивных алгоритмов, основное достоинство которых – это простота и лаконичность исходного кода, за что часто приходится платить повышенными требованиями к машинным ресурсам. Реальные экономические задачи требуют использования рекурсивных методов решения при большой глубине вложенности дерева рекурсии, что приводит к значительным временным затратам. В этом аспекте практический интерес представляет получение достаточно точных оценок времени выполнения рекурсивных алгоритмов.

Основной особенностью анализа ресурсной эффективности рекурсивных алгоритмов является необходимость учета дополнительных затрат памяти и трудоемкости, связанных с механизмом организации рекурсии в принятой модели вычислений. Получение ресурсных функций вычислительных алгоритмов в рекурсивной

реализации базируется как на методах оценки вычислительной сложности собственно тела рекурсивного алгоритма, так и на способах учета ресурсных затрат на организацию рекурсии и детальном анализе цепочки рекурсивных вызовов и возвратов — порожденного дерева рекурсии. Трудоемкость рекурсивных реализаций алгоритмов, очевидно, связана как с количеством операций, выполняемых при одном вызове рекурсивной функции, так и с количеством таких вызовов. Должны быть учтены также и затраты на возврат вычисленных значений и передачу управления в точку вызова. Эти операции должны быть включены применяемую модель вычислений, и, следовательно, — в функцию трудоемкости рекурсивно реализованного алгоритма.

Анализ трудоемкости вызова рекурсивной функции. Механизм вызова функции или процедуры в языке высокого уровня существенно зависит от архитектуры компьютера и операционной системы. В рамках архитектуры и операционных систем IBM PC совместимых компьютеров этот механизм реализован через программный стек. Как передаваемые в процедуру или функцию фактические параметры, так и возвращаемые из них значения, помещаются в программный стек специальными командами процессора — мы считаем эти операции базовыми в рассматриваемой модели вычислений. Для подсчета трудоемкости вызова в базовых операциях обслуживания стека, напомним, что при вызове процедуры или функции в стек помещается адрес возврата, состояние необходимых регистров процессора, состояние локальных ячеек вызывающей процедуры или функции, адреса возвращаемых значений и передаваемые параметры. После этого выполняется переход по адресу на вызываемую процедуру, которая извлекает переданные фактические параметры, выполняет вычисления и помещает результаты по указанным в стеке адресам. При завершении работы вызываемая процедура восстанавливает регистры, локальные ячейки, выталкивает из стека адрес возврата и осуществляет переход по этому адресу. Для анализа трудоемкости механизма вызова-возврата для рекурсивной процедуры введем следующие обозначения: p — количество передаваемых фактических параметров, r — количество сохраняемых в стеке регистров, k — количество возвращаемых по адресной ссылке значений, l — количество локальных ячеек процедуры. Поскольку каждый объект в некоторый мо-

мент помещается в стек, и в какой-то момент выталкивается из него, то трудоемкость рекурсивной процедуры, связанная с обслуживанием одного вызова и одного возврата, обозначаемая в дальнейшем через $f_R(\_)$ составит в базовых операциях принятой модели вычислений:

$$f_R(\_) = 2 \cdot (p + k + r + l + \_). \quad (1)$$

Дополнительная единица в формуле (1) учитывает операции с адресом возврата. Для рекурсивной функции мы должны дополнительно учесть еще одну локальную ячейку, через которую собственно передается значение функции. Мы обозначим этот параметр через f, считая, что его значение равно единице для реализации в виде функции и нулю — для реализации в виде процедуры. Тогда трудоемкость на один вызов-возврат рекурсивной функции может быть определена следующим образом:

$$f_R(\_) = 2 \cdot (p + k + r + f + l + \_)f = \quad . \quad (2)$$

Анализ трудоемкости рекурсивных алгоритмов в части совокупной трудоемкости самого рекурсивного вызова-возврата можно выполнять разными способами в зависимости от того, как формируется итоговая сумма базовых операций — либо отдельно по цепочкам рекурсивных вызовов и возвратов, либо совокупно по вершинам дерева рекурсии. Формулы (1) и (2) отражают второй способ, заметим также, что если положить $f = 0$ для рекурсивной процедуры в формуле (2), то она сводится к (1).

Учет особенностей рекурсивной реализации в функциях ресурсной эффективности программных реализаций алгоритмов. Суммарные ресурсы, требуемые рекурсивной реализацией алгоритма, могут быть разделены на ресурсы, собственно связанные с решением задачи, и ресурсы, необходимые для организации рекурсии. При сравнительном анализе итерационной и рекурсивной реализации можно отметить, что особенность последней состоит как в наличии дополнительных операций, организующих рекурсию, так и дополнительных затрат оперативной памяти в области программного стека, для хранения информации о цепочке рекурсивных вызовов. При этом отметим, что эти затраты в ряде случаев оправданы, например, рекурсивные алгоритмы, использующие метод декомпозиции, позволяют получить асимптотически более эффективные алгоритмы для це-

лого ряда задач [1, 2, 3]. Эта эффективность достигается в частности и за счет большего объема памяти в области программного стека, что должно быть обязательно учтено в комплексном критерии оценки качества алгоритмов.

Оценка требуемой памяти в стеке может быть получена следующим образом: поскольку рекурсивные вызовы обрабатываются последовательно, то во временной динамике в области стека хранится не полный фрагмент дерева рекурсии, а только текущая цепочка рекурсивных вызовов — унарный фрагмент этого дерева. Из этого следует, что требуемый объем памяти в области программного стека определяется не общим количеством вершин порождённого дерева рекурсии, а максимальной глубиной его листьев. Очевидно, что вид и глубина рекурсивного дерева определяются как особенностями самого алгоритма, так и характеристиками множества исходных данных. Обозначив через $H_R(D)$ максимальную глубину рекурсивного дерева, порождаемого данным алгоритмом на данном входе D, можно оценить требуемый объем программного стека, опираясь на реализацию механизма вызова функции. Предполагая, в худшем случае, что параметры, передаваемые через стек, сохраняются в нем, получаем, что максимальный объем памяти в области стека может быть определен на основе (2) следующим образом

$$V_{st}(D) = H_R(D) \cdot (p + \cdot + z + f + \ +1) \cdot l_w, \qquad (3)$$

где l_w — длина слова в байтах.

Отметим также, что в отличие от объема памяти в области программного стека, требуемого для организации рекурсии, который зависит от максимальной глубины рекурсивного дерева, количество операций со стеком на один вызов-возврат, задаваемое формулой (2), должно быть учтено в функции трудоемкости для всех вызовов. Таким образом, получение функции трудоемкости в рекурсивной реализации требует определения общего количества вершин рекурсивного дерева. Если структура дерева такова, что оно является не только глубоким, но и «широким», то совокупные затраты на организацию рекурсии могут быть значительны. Трудоемкость тела рекурсивной функции или процедуры может быть получена на основе методов анализа итерационных алгоритмов (см., например, [4, 5]). Однако общая трудоемкость определяется числом порожденных вершин дерева рекурсии, кроме того, мы должны учесть, что в общем случае трудоемкость в

разных вершинах может различаться, и зависеть от данных или параметров рекурсии.

Анализ трудоемкости методом подсчета вершин дерева рекурсии. При теоретическом построении ресурсных функций рекурсивного алгоритма необходимо учесть ряд ресурсных затрат и особенностей рекурсивной реализации, а именно:

— ресурсные затраты на обслуживание рекурсивных вызовов-возвратов, передачу параметров и возврат значений рекурсивных функций (ресурсные затраты обслуживания рекурсии);

— специфику фрагмента останова рекурсии, приводящую к необходимости отдельного учета ресурсных затрат в листьях дерева рекурсии.

Учет этих особенностей при теоретическом анализе рекурсивных алгоритмов приводит к необходимости получить зависимости общего числа вершин дерева рекурсии и числа его внутренних вершин и листьев, как функции от характеристик множества входных данных. Если мы можем определить ресурсные затраты в каждой вершине дерева, то суммируя мы получим ресурсную функцию алгоритма в целом. Такой подход мы и будем называть в дальнейшем *методом получения ресурсных функций рекурсивных алгоритмов на основе анализа порожденного дерева рекурсии*

Анализ дерева рекурсии. Первым этапом предлагаемого метода является исследование структуры порождённого дерева рекурсии. В предположении, что n — длина входа алгоритма для классов N, NPR [4]; $m_1,\cdots m_k$ — значение параметров входа для классов PR, NPR ($k \le \imath$) [4], а D — конкретный вход алгоритма, введем следующие обозначения для характеристик дерева рекурсии порождённого рекурсивным алгоритмом на входе D:

$R(D)$ — общее количество вершин дерева рекурсии,

$R_V(D)$ — количество внутренних вершин дерева,

$R_L(D)$ — количество листьев дерева рекурсии,

$H_R(D)$ — максимальная глубина дерева рекурсии (максимальное по всем листьям дерева количество вершин в пути от корня дерева до листа).

Тогда очевидно, что справедливы соотношения

$$R(D) = R_V(D) + R_L(D),\ H_R(D) \le R_V(D) + \ .$$

Таким образом, основная задача при использовании этого метода состоит в теоретическом построении функций $R_V(D)$, $R_L(D)$ и $H_R(D)$ — как функций от характеристик множества входных данных в зависимости от принадлежности алгоритма к одному из основных классов. Дополнительный интерес, в смысле анализа дерева рекурсии, представляет информация об отношении количества листьев к общему количеству вершин рекурсивного дерева. Эта характеристика относительной «ширины» нижнего уровня (уровня листьев) в дереве рекурсии

$$B_L(D) = \frac{R_L(D)}{R(D)}, \quad 0 < B_L(D) < \ . \tag{4}$$

Значение $B_L(D)$ будет минимально для цепочки (унарного дерева), и будет возрастать при увеличении числа вершин, порожденных во внутренних вершинах дерева рекурсии.

Рассмотрим более подробно, как на основе функций $R_V(D)$ и $R_L(D)$ возможно теоретическое построение ресурсных функций рекурсивных алгоритмов.

Получение функции трудоемкости методом анализа порождённого дерева рекурсии. Для рекурсивных алгоритмов трудоемкость решения конкретной задачи включает в себя не только трудоемкость непосредственной обработки данных в теле рекурсивной функции, но и затраты на организацию рекурсии. Более точно, трудоемкость алгоритма A на конкретном входе D — $f_A(D)$ определяется трудоемкостью обслуживания дерева рекурсии, зависящей от общего количества его вершин, и трудоемкостью продуктивных вычислений, выполненных во всех вершинах дерева рекурсии. В связи с этим обозначим через

$f_R(D)$ — трудоемкость порождения и обслуживания дерева рекурсии,

$f_C(D)$ — трудоемкость продуктивных вычислений алгоритма,

и с учетом введенных обозначений получаем

$$f_A(D) = f_R(D) + f_C(D). \tag{5}$$

Трудоемкость обслуживания дерева рекурсии может быть вычислена достаточно просто, а именно, если функция $R(n)$ известна, и на обслуживание одного рекурсивного вызова затрачивается фиксированное количество базовых операций — $f_R(n)$, определяемых по формуле (2), то

$$f_R(n) = R(n) \cdot f_R(n). \quad (6)$$

При подсчете трудоемкости продуктивных вычислений необходимо учесть, что для листьев рекурсивного дерева алгоритм будет выполнять непосредственное вычисление значений, и эта трудоемкость отлична от трудоемкости во внутренних вершинах, следовательно, трудоемкость алгоритма при останове рекурсии должна быть учтена отдельно. В связи с этим обозначим через

$f_{CV}(n)$ — трудоемкость продуктивных вычислений (обработки данных) во внутренних вершинах,

$f_{CL}(n)$ — трудоемкость вычислений в листьях дерева рекурсии.

Эти обозначения позволяют представить функцию $f_C(n)$ в виде

$$f_C(n) = f_{CV}(n) + f_{CL}(n). \quad (7)$$

Обозначим через $f_{CL}(n)$ трудоемкость алгоритма при останове рекурсии — это есть трудоёмкость в одном листе порождённого дерева. Заметим, что, как правило, значение $f_{CL}(n)$ может быть сравнительно легко получено, т. к. для большинства задач, особенно при рекурсивной реализации метода декомпозиции [3] выражается фиксированным числом базовых операций. Зная количество листьев порождённого дерева рекурсии, можно определить $f_{CL}(n)$

$$f_{CL}(n) = R_L(n) \cdot f_{CL}(n). \quad (8)$$

Во внутренних вершинах дерева выполняются некоторые действия, связанные с подготовкой параметров следующих рекурсивных вызовов и обработкой возвращаемых результатов. Трудоемкость такой обработки может зависеть как от обрабатываемых в этой вершине данных, так и от положения вершины в дереве рекурсии. С целью учета этой зависимости, введем нумерацию внутренних вершин, начиная с корня, по уровням дерева. Заметим, что число уровней внутренних вершин в дереве на единицу меньше глубины рекурсии $H_R(n)$. Пусть m есть номер уровня $m = \overline{1, H_R(n)-1}$, а k — номер вершины на данном уровне $k = \overline{1, K(m)}$,

где $K(n)$ — количество внутренних вершин на уровне m. Заметим, что неполное дерево на уровне k может содержать как внутренние вершины, так и листья. С учетом такой нумерации обозначим вершины дерева через

$$v_{mk},\ m = \overline{1, H_R(D)-1};\ k = \overline{1, K(n)},$$

при этом очевидно, что

$$\sum_{m=1}^{H_R(D)-1} \sum_{k=1}^{K(n)} 1 = \Psi_V(D).$$

Обозначим трудоемкость продуктивных вычислений в вершине v_{mk} через $f_{CV}(v_{mk})$, тогда формула для трудоемкости продуктивных вычислений во внутренних вершинах дерева рекурсии имеет вид

$$f_{CV}(D) = \sum_{m=1}^{H_R(D)-1} \sum_{k=1}^{K(n)} f_{CV}(v_{mk}). \qquad (9)$$

Заметим, что в случае, когда значения функции $f_{CV}(v_{mk})$ не зависят от номера вершины дерева рекурсии, т. е. трудоемкость продуктивных вычислений в вершинах не зависит от данных, то, обозначая трудоемкость продуктивных вычислений для любой внутренней вершины дерева через $f_{CV}(v)$, имеем

$$f_{CV}(D) = \Psi_V(D) \cdot f_{CV}(v). \qquad (10)$$

Подставляя полученные результаты в формулу (5), окончательно получаем формулы для определения трудоемкости рекурсивного алгоритма на основе метода анализа порождённого дерева рекурсии. В общем случае, при котором трудоемкость продуктивных вычислений различна во внутренних вершинах дерева

$$f_A(D) = R(D) \cdot f_R(v) + R_L(D) \cdot f_{CL}(v) + \sum_{m=1}^{H_R(D)-1} \sum_{k=1}^{K(n)} f_{CV}(v_{mk}), \qquad (11)$$

и в частном случае, когда трудоемкость продуктивных вычислений для любой внутренней вершины дерева рекурсии одинакова

$$f_A(D) = \Psi(D) \cdot f_R(v) + \Psi_L(D) \cdot f_{CL}(v) + \Psi_V(D) \cdot f_{CV}(v). \qquad (12)$$

По аналогии с характеристикой $B_L(D)$ можно рассмотреть дополнительную характеристику трудоемкости рекурсивного алгоритма — долю операций обслуживания дерева рекурсии. Обозначая ее через $F_R(D)$ имеем

$$F_R(n) = \frac{f_R(n)}{f_A(n)} \quad 0 < F_R(n) < .$$

Значение $F_R(n)$ показывает насколько трудоемкость обслуживания дерева рекурсии значима в общей трудоемкости рекурсивного алгоритма.

Заключение. В заключение сформулируем этапы анализа трудоемкости рекурсивного алгоритма методом анализа порождённого дерева рекурсии

1. Анализ порождаемого данным алгоритмом дерева рекурсии с целью получения теоретических зависимостей для характеристик дерева — $R(n, m_1,\cdots m_k)$, $R_L(n, m_1,\cdots m_k)$, $R_V(n, m_1,\cdots m_k)$, $H_R(n, m_1,\cdots m_k)$, как функций от длины входа (n) и/или характеристических особенностей множества входных данных.

2. Определение трудоемкости обслуживания рекурсии на одни вызов-возврат — $f_R(n)$ по формуле (2).

3. Определение трудоемкости алгоритма при останове рекурсии — $f_{CL}(n)$. Если останов происходит при нескольких значениях аргумента, и трудоемкость вычисления в разных листьях различна, то необходим более детальный подсчет, но уже с учетом типов листьев в дереве рекурсии.

4. Исследование трудоемкости продуктивных вычислений во внутренних вершинах дерева рекурсии — получение функции $f_{CV}(v_{mk})$.

5. Получение функции трудоемкости рекурсивного алгоритма по формулам (11) или (12) в зависимости от поведения функции $f_{CV}(v_{mk})$.

Особо отметим, что применение предложенного метода к входам, обуславливающим лучший и худший случай трудоёмкости для алгоритмов количественно-параметрического класса позволяет получить функции трудоёмкости исследуемого алгоритма в лучшем и худшем случая соответственно, как функции длины входа алгоритма.

Список литературы

[1.] Ахо А., Хопкрофт Дж., Ульман Дж. Построение и анализ вычислительных алгоритмов: Пер. с англ.: — М.: Мир, 1979. — 546 с.

[2.] Ахо А., Хопкрофт Дж., Ульман Дж. Структуры данных и алгоритмы: Пер. с англ.: — М.: Издательский дом «Вильямс», 2001. — 384 с.

[3.] Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ, 2-ое издание: Пер. с англ. — М.: Издательский дом «Вильямс», 2005. — 1296 с.

[4.] Ульянов М. В. Классификация и методы сравнительного анализа вычислительных алгоритмов. Научное издание. — М.: Издательство физико-математической литературы, 2004. — 212 с.

[5.] Ульянов М. В. Дополнение к книге Дж. Макконелла «Основы современных алгоритмов». — М.: Издательство Техносфера, 2004. С. 303–366.