УДК 004.4'24+519.6

Автоматизация анализа вычислительной и емкостной сложности алгоритмов на множествах и графах

© Г.С. Иванова

МГТУ им. Н.Э. Баумана, Москва, 105005, Россия

Предложен подход, позволяющий автоматически получить оценки вычислительной, временной и емкостной сложности структурных алгоритмов решения задач структурного анализа и синтеза, описанных в операциях над графами и/или множествами. Алгоритм реализует метод D-карт.

Алгоритм оценки вычислительной и временной сложности предполагает рекурсивное распознавание базовых и производных структурных конструкций алгоритма с применением их инвариантов, расчет интегральных характеристик этих конструкций и свертку распознанных конструкций с назначением им соответствующих оценок. Оценка емкостной сложности алгоритма выполняется с использованием моделей задействованных структур данных.

Ключевые слова**:** *задачи структурного синтеза, структурный алгоритм, вычислительная сложность, емкостная сложность, автоматизация оценки.*

Введение. Алгоритмы решения комбинаторных задач структурного анализа и синтеза часто имеют экспоненциальную оценку вычислительной сложности, и при больших размерностях входов их выполнение может потребовать нереализуемо больших затрат временных и емкостных ресурсов вычислительной системы. Поэтому в процессе разработки алгоритма необходимо проводить анализ его вычислительной и емкостной сложности.

Исследованию свойств алгоритмов решения задач рассматриваемого класса уделяется большое внимание, наиболее известными в этой области являются работы [1, 2]. При этом анализ вычислительной и емкостной сложности алгоритмов их разработчики проводят вручную.

В настоящее время в литературе встречается единственное упоминание о системе, выполняющей автоматизированный анализ алгоритмов на графах, в которой для оценки вычислительной сложности использован вероятностный подход [3]. Он имеет ряд недостатков. Во-первых, оценка осуществляется по программе на алгоритмическом языке и, соответственно, жестко к нему привязана. Во-вторых, класс алгоритмов, для которых выполняется оценка, ограничен заданным в системе набором расчетных функций, что существенно сужает область применения системы. И, наконец, в-третьих, система выдает только асимптотическую оценку, в то

время как при относительно больших размерностях входа задачи объективную картину дает только функциональная оценка.

Предлагаемый подход лишен перечисленных недостатков.

Основные принципы и математические соотношения, положенные в основу автоматизации расчета. В основу автоматизации расчета временной сложности алгоритма положен метод непосредственного расчета функции сложности выполнения программы от размерности входа, применимый к структурным алгоритмам. Этот метод, часто называемый методом *D*-карт [3], вручную использует большинство авторов, занимающихся разработкой и анализом алгоритмов.

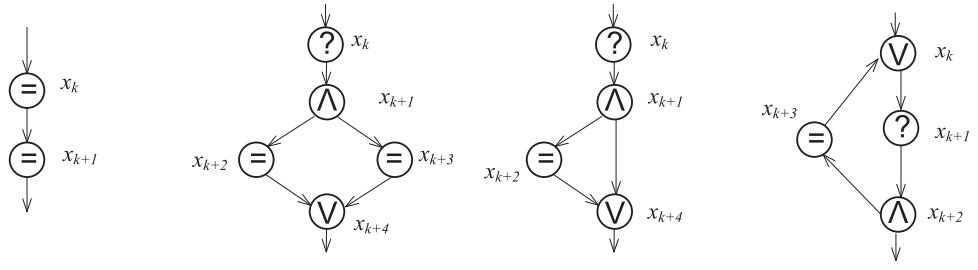
Исходными данными для получения функций вычислительной сложности от входа задачи являются время выполнения операторов в условных операциях или тактах и количество повторений каждого оператора, которое определяется размером входа задачи и структурой алгоритма. Функция временной сложности отличается от функции вычислительной сложности точностью расчета характеристик времени выполнения. Если характеристика определяется с точностью до количества операторов некоторого типа, как правило операторов сравнения, то получаем функцию вычислительной сложности. Если характеристика определяется во временных единицах, то получаем функцию временной сложности.

Для обеспечения универсальности оценки анализ характеристик производительности алгоритма должен выполняться по его модели с использованием характеристик области определения входных данных и области интерпретации этой модели [4]. Соответственно точность оценки зависит от степени детализации функций и предикатов области интерпретации.

Поскольку метод непосредственного расчета применим только для структурных алгоритмов, вычисления будем выполнять по рассмотренной в [5] модели структурного алгоритма, которая построена на базе предложенной в [4] более общей модели.

Структурный алгоритм представляет собой совокупность базовых (Следование, Ветвление, Цикл-пока) и производных конструкций, которые строятся из базовых в соответствии с принципами структурности и являются иерархически сложными с некоторым количеством уровней вложения [6].

Модель структурного алгоритма, предложенная в [5], базируется на моделях базовых структурных конструкций G^1, G^2, G^3 (рис. 1) и модели оператора изменения данных G^0. При этом модели производных структурных конструкций также строятся из базовых с использованием правил иерархического вложения.



а *б* *в*

Рис. 1. Модели базовых структурных конструкций:
а — Следование (G^1); *б* — Ветвление (G^2); *в* — Цикл-пока (G^3);

(=) – обработка данных; (∧) – ветвление потоков управления;
(?) – вычисление условий; (∨) – слияние потоков управления

Формальное описание структуры структурного алгоритма и правила разбора таких алгоритмов определяет аксиоматика операции свертки *factor( )* над сложными структурными конструкциями, также описанная в [5]:

$$factor(G^0) = G^{0F} = G^0,$$
$$factor(G_i \in \{G^1, G^{1F}, G^2, G^{2F}, G^3, G^{3F}\}) = G^{0F},$$
$$factor(\{G_k\} \subset G^{1C} \;\&\; G_k \in \{G^{2F}, G^{3F}\}) = G^{1F},$$
$$factor(G_{01}, G_{02} \subset G^{2C} \;\&\; G_{01}, G_{02} \in \{G^1, G^{1F}, G^2, G^{2F}, G^3, G^{3F}\}) = G^{2F},$$
$$factor(G_0 \subset G^{3C} \;\&\; G_0 \in \{G^1, G^{1F}, G^2, G^{2F}, G^3, G^{3F}\}) = G^{3F},$$

где G^{1C} — кусок управляющего графа, соответствующий линейной последовательности операторов обработки данных, операторов Ветвления и операторов Цикла-пока (в свою очередь, возможно, включающих внутренние структурные конструкции), т. е. $G^{1C} = Ch(G_i, G_j)$. Здесь *Ch* — последовательность кусков, являющихся моделями указанных выше последовательных операторов, причем

$$(\forall k = \overline{i, j}\,)\; G_k \in \{G^0, G^2, G^{2F}, G^3, G^{3F}\};$$

G^{2C} — кусок управляющего графа, соответствующий оператору ветвления (сложной структурной конструкции типа G^2), ветви которого $G_{01}, G_{02} \subset G^{2C}$ могут содержать вложенные структурные конструкции, т. е. $G_{01}, G_{02} \in \{G^1, G^{1F}, G^{2F}, G^3, G^{3F}\}$; G^{3C} — кусок управляющего графа, соответствующий оператору Цикла-пока (структурной конструкции типа G^3),

тело которого $G_0 \subset G^{3C}$ может содержать вложенные структурные конструкции, т. е. $G_0 \in \{G^1, G^{1F}, G^{2F}, G^3, G^{3F}\}$.

В итоге получаем множество моделей конструкций, изоморфных базовым G^B и формируемых из сложных посредством свертки:

$$G^{BF} = \{G^{0F}, G^{1F}, G^{2F}, G^{3F}\}.$$

Таким образом, процесс анализа модели исследуемого алгорит-ма заключается в ее декомпозиции на фрагменты, *гомеоморфные* или, после факторизации вложенных конструкций, изоморфные моделям базовых конструкций. Распознавание вида конструкций перед их сверткой осуществляется на основе выявленных в [5] инвариантов вида куска.

Модель каждой структурной конструкции, а также модель оператора изменения данных сопоставим с оценкой вычислительной или, соответственно, временной сложности ее выполнения. Вычисление интегральной оценки будем осуществлять в процессе факторизации структурных конструкций при их распознавании.

Модели G^0 оператора изменения данных поставим в соответствие его вычислительную Q^0 или временную сложность T^0, которые определяются количеством операций сравнения элементов структур данных или временем выполнения этих операций.

При свертке конструкции G^1 (Следование) функции временной и вычислительной сложности входящих в нее операторов G^{01} и G^{02} складываются

$$Q^1 = Q^{01} + Q^{02}, \; T^1 = T^{01} + T^{02}. \qquad (1)$$

Функции вычислительной и временной сложности Ветвления зависят от количества проходов по каждой ветви. Обычно это учитывают, вводя вероятности выбора ветвей. Однако расчет этих вероятностей, как правило, задача не тривиальная, поскольку они зависят от значений исходных данных. Поэтому при свертке конструкции G^2 будем считать вычислительную и временную сложность как полусумму сложностей ветвей G^{01} и G^{02}:

$$Q^2 = (Q^{01} + Q^{02}) / 2, \quad T^2 = (T^{01} + T^{02}) /2. \qquad (2)$$

В алгоритмах решения задач структурного анализа и синтеза в основном встречаются счетные циклы, количество повторений которых известно либо, если анализируется подмножество универсума, сравнительно просто можно оценить «сверху». Так, если в цикле перебираются значения некоторой целочисленной переменной — $\forall i = 1, k$, то количество его повторений, очевидно, равно k. Если в цикле операции выполняются для всех элементов некоторого подмножества — $\forall x \in X1$, то количество его повторений при условии, что

$X1 \subset X$, $|X| = n$, можно оценить «сверху», как $| X1| \approx n$. И, наконец, для варианта с дополнительным условием — $\forall x \in X1$ & <Условие> в качестве оценки «сверху» также применимо значение $|X1| \approx n$.

В некоторых случаях мощность подмножества универсума может быть ограничена физическим смыслом моделируемой системы, например заданным количеством выводов элементов ρ или нагрузочной способностью элемента A. Использование этих величин при оценке количества повторений цикла позволяет существенно уточнить оценки вычислительной и временной сложности. Информацию о подобных ограничениях количества повторений цикла целесообразно получать от разработчика алгоритма через специальный формат описания алгоритма или в виде ответов пользователю на соответствующие запросы.

Функции вычислительной и временной сложности счетного цикла определяются умножением функций сложности тела цикла на количество его повторений k:

$$Q^3 = kQ^{01}, \; T^3 = kT^{01}. \qquad (3)$$

Однако кроме счетных циклов в алгоритмах задач структурного анализа и синтеза встречаются и итерационные циклы, количество повторений которых неизвестно, так как зависит от конкретных данных. Например, рассмотрим алгоритм решения задачи дихотомического разрезания по методу ветвей и границ. В пределе при выполнении этого алгоритма может осуществляться полный перебор. Более точно вычислительную сложность подобных алгоритмов на этапе компиляции без информации о конкретных наборах данных оценить невозможно. В этом случае также целесообразно запросить оценку количества повторений итерационного цикла у пользователя, поскольку оно может ограничиваться конструктивными особенностями рассматриваемых объектов.

При свертке конструкций, состоящих из факторизованных элементов, вычислительная и временная сложности также должны рассчитываться по формулам (1)–(3).

Методика расчета вычислительной, временной и емкостной сложности алгоритма. Определение вычислительной, временной и емкостной сложности структурного алгоритма по описанию в операциях над графами состоит из следующих действий [8–10].

1. По описанию алгоритма в операциях над графами

$$A = A(OP(G_1), OP(G_2), \ldots, OP(G_r)),$$

определить множество операций $OP(G_i)$ над каждым графом G_i:

$$\{\{Op_j(G_i) / j = 1, J_i\} / i = 1, r\},$$

где $Op_j(G_i)$ — j-я операция над графами G_i, а J_i — количество таких операций.

2. Используя библиотеку описания операций над графами с учетом выбранного способа представления, определить множество операций Op_t над каждым множеством $M_{i,p}$, $p = 1, P_i$, представляющим граф G_i:

$$\{\{\{Op_t(M_{i,p}) / t = 1, T_{i,p}\} / p = 1, P_i\} / i = 1, r\},$$

где $Op_t\left(M_{i,p}\right) = \bigcup_{j=1}^{J_i} Op_{t,j}\left(M_{i,p}\right)$ — совокупность t-х операций над p-м множеством, представляющим i-й граф, которая выполняется при реализации всех J_i операций над этим графом.

3. Построить модели структур данных S_i [7], выбранных для представления совокупности множеств $M_{i,p}$ каждого графа G_i, и определить их емкостную сложность $L(S_i)$:

$$\{L(S_i) / i = 1, r\}.$$

4. Используя те же модели, определить функциональную вычислительную сложность $Q_d(S_i)$ выполнения основных операций $Op_d(S_i)$ над элементами структуры данных S_i:

$$\{\{Q_d(S_i) / d = 1, D\} / i = 1, r\},$$

где D — мощность множества основных операций.

5. Построить модель выполнения каждой операции $Op_t(M_{i,p})$ над множеством $M_{i,p}$ в основных операциях над элементами структуры данных $Op_d(S_i)$ и, используя полученные в п. 4 вычислительные сложности реализации основных операций над элементами структур данных $Q_d(S_i)$ и формулы (1)–(3), получить функциональную вычислительную сложность ее выполнения $Q_t(M_{i,p})$, т. е. определить

$$\{\{\{Qt(M_{i,p}) / t = 1, T_{i,p}\} / p = 1, P_i\} / i = 1, r\}.$$

6. Построить модель выполнения каждой операции $Op_j(G_i)$ над графом G_i в элементарных операциях над множествами $Op_t(M_{i,p})$ и, используя полученные в п. 5 вычислительные сложности реализации элементарных операций над множествами и (1)–(3), получить функциональную сложность ее выполнения $Q_j(G_i)$, т. е. определить

$$\{\{Q_j(G_i) / j = 1, J_i\} / i = 1, r\}.$$

7. Построить модель алгоритма в элементарных операциях над графами и, используя формулы (1)–(3), получить функциональную вычислительную сложность этого алгоритма Q_A.

8. Рассчитать емкостную сложность алгоритма L_A как сумму

емкостных сложностей структур данных, представляющих графовые модели G_i:

$$L_A \sum_{i=1}^{r} L(S_i).$$

Поскольку вычислительная и временная сложности по определению пропорциональны, расчетные соотношения пп. 4–7 позволяют также оценить время выполнения программы, реализующей разрабатываемый алгоритм. В качестве исходных данных для такого расчета следует вместо функциональной вычислительной сложности $Q_d(S_i)$ (см. п. 4) использовать оценки времени выполнения операций над структурой данных, реализующей i-й граф — $T_d(\mathrm{S}_i)$.

Полученные по перечисленным выше расчетным соотношениям значения вычислительной, временной и емкостной сложности должны проверятся разработчиком алгоритмов на допустимость. В том случае, если время или объем используемой памяти не соответствует требованиям, разработчик должен оптимизировать разрабатываемый алгоритм или использовать другой метод решения задачи.

Заключение. Предложенные в настоящей работе расчетные соотношения и методика оценки вычислительной, временной и емкостной сложности алгоритмов решения задач структурного анализа и синтеза позволяют автоматизировать трудоемкий процесс анализа разрабатываемых алгоритмов и получать более точные оценки наиболее существенных временных и емкостных характеристик этих алгоритмов. В результате сокращается время разработки программ решения задач структурного анализа и синтеза и появляется возможность улучшения их качества.

ЛИТЕРАТУРА

[1] Кормен Т., Лейзерсон Ч., Риверст Р., Штайн К. *Алгоритмы: построение и анализ*. Москва, Вильямс, 2005, 1296 с.

[2] Кнут Д.Э. *Искусство программирования*. Москва, Наука, 2006, 437 с.

[3] Vitter J., Flajolet Ph. Average-Case Analyse of Algorithms and Data Structure. *Handbook of Handbook of theoretical computer science: algorithms and complexity*, 1991, vol. A, pp. 431–524.

[4] Овчинников В.А., Иванова Г.С. Информационно-логическая модель алгоритма. *Вестник МГТУ им. Н.Э. Баумана*. *Сер. Приборостроение*, 2005, № 2(59), с. 109–121.

[5] Иванова Г.С. Формальная постановка задачи структуризации алгоритмов. *Вестник МГТУ им. Н.Э. Баумана. Сер. Приборостроение*, 2005, № 3 (60), с. 64–73.

[6] Вирт Н. *Алгоритмы и структуры данных. Новая версия для Оберона*. Москва, ДМК Пресс, 2010, 406 с.

[7] Иванова Г.С. Математические модели структур данных. *Информационные технологии*, 2006, № 9, с. 44–52.

[8] Овчинников В.А. Операции над ультра- и гиперграфами для реализации процедур анализа и синтеза структур сложных систем. Ч. 1. *Инженерное образование*, 2009, № 10. URL: http://technomag.bmstu.ru/doc/132769.html

[9] Овчинников В.А. Операции над ультра- и гиперграфами для реализации процедур анализа и синтеза структур сложных систем. Ч. 2. *Инженерное образование*, 2009, № 11. URL: http://technomag.bmstu.ru/doc/133223.html

[10] Овчинников В.А. Операции над ультра- и гиперграфами для реализации процедур анализа и синтеза структур сложных систем. Ч. 3. *Инженерное образование,* 2009, № 12. URL: http://technomag.bmstu.ru/doc/134335.html

Статья поступила в редакцию 24.06.2013 г.

Ссылку на эту статью просим оформлять следующим образом:

Иванова Г.С. Автоматизация анализа вычислительной и емкостной сложности алгоритмов на множествах и графах. *Инженерный журнал: наука и инновации*, 2013, вып. № 11. URL:http://engjournal.ru/catalog/it/hidden/1043.html

Иванова Галина Сергеевна родилась в 1954 г., окончила МВТУ им. Н.Э. Баумана в 1978 г. Д-р техн. наук, профессор кафедры «Компьютерные системы и сети» МГТУ им. Н.Э. Баумана. Автор свыше 50 научных работ в области вычислительной техники. Специализируется в области проектирования программных систем. e-mail: gsivanova@gmail.com