Классификация методов построения алгоритмических систем

М. В. Ульянов, О. А. Наумова
Московский государственный университет печати, Россия
e-mail: muljanov@mail.ru, helga_13_07@mail.ru

Предложена оригинальная классификация методов построения алгоритмических систем, в основу которой положены возможные результаты кластеризации и последующего распознавания исходных и промежуточных данных алгоритма. Предложенная классификация дает новый подход к построению алгоритмических систем, состоящий в том, что при достаточно надежной кластеризации входов алгоритма возможны назначение рациональных алгоритмов для устойчивых кластеров входных данных и выбор алгоритма для текущего входа по результату решения задачи распознавания этого входа в многомерном пространстве параметров.

Ключевые слова: алгоритмические системы, алгоритмические комплексы, комбинированные алгоритмы, классификация методов построения.

Введение

Задача оценки эффективности алгоритма традиционно понимается как задача оценки его качества с помощью некоторой функции, аргументом которой является длина входа. Компоненты такой функции — в первую очередь ресурсные характеристики, т. е. трудоемкость и требуемый объем дополнительной памяти. Функция оценки качества может включать в себя и ряд других компонент, отражающих требования проблемной области применения и специфику решаемой задачи, например, точность получаемого алгоритмом решения, скорость сходимости и т. д. [1].

Рассмотрение оценки качества алгоритма без учета особенностей задачи и специфики ее применения приводит к фиксации только ресурсных характеристик алгоритма. Для сохранения общности изложения будем далее считать, что функция качества алгоритма определяется его ресурсными характеристиками, а алгоритм решения некоторой задачи, обладающий лучшими ресурсными характеристиками, назовем *ресурсно-эффективным* в рамках рассматриваемой совокупности алгоритмов [2]. Отметим в этой связи, что в настоящее время несмотря на возрастающие мощности современных компьютеров такая характеристика алгоритмов как временная эффективность, или трудоемкость, остается важной для целого ряда задач и областей применения, в том числе для точных алгоритмов решения NP-трудных задач.

При создании ресурсно-эффективного алгоритмического обеспечения программных средств и систем могут быть использованы разнообразные пути. Прежде всего назовем наиболее трудный, но теоретически наиболее значимый путь — создание новых алгоритмов решения данной задачи; другим путем является модификация существующих алгоритмов на основе их тщательного анализа и исследования в различных классах исходных данных [2]. Наконец, назовем еще один путь повышения ресурсной эффективно-

сти алгоритмического обеспечения — создание алгоритмических систем, опирающееся на существующее множество алгоритмов решения актуальных задач.

В настоящее время для большинства задач компьютерной математики известен целый ряд различных алгоритмов, доставляющих правильное решение данных задач и обладающих при этом различными ресурсными характеристиками, зависящими от особенностей входных данных [3]. За последние десятилетия портфель алгоритмов компьютерной математики пополнился рядом эффективных алгоритмов решения актуальных задач. В качестве примеров можно привести асимптотически оптимальный алгоритм Штрассена—Шенхаге умножения длинных целых чисел [4], алгоритмы решения задач факторизации и дискретного логарифмирования [5], новые оригинальные алгоритмы сортировки [6], ряд алгоритмов для решения специальных постановок задач целочисленного программирования [7], ε-полиномиальные алгоритмы решения NP-трудных задач [3] и т. д.

Существующее алгоритмическое разнообразие создает базу для построения алгоритмических систем — объединенных некоторым образом различных алгоритмов решения рассматриваемой задачи. При этом использование в полученной алгоритмической системе алгоритмов, наиболее рациональных для соответствующих условий применения, гарантирует в общем случае лучшие ресурсные характеристики всей системы в целом по сравнению с отдельными входящими в нее алгоритмами [2]. Разработка алгоритмических систем является сегодня одним из перспективных направлений совершенствования алгоритмической поддержки программного обеспечения.

В рамках вышесказанного представляет интерес анализ и систематизация различных методов построения алгоритмических систем с их последующей классификацией — рассмотрению этих вопросов и посвящена данная статья. Объектом исследования являются алгоритмические системы, предметом исследования — различные методы построения алгоритмических систем в аспекте их классификации.

1. Терминология и обозначения

Следуя в основном работам [2, 8], будем использовать следующую терминологию и обозначения, связанные с проблематикой алгоритмов и анализом их ресурсной эффективности:

Z — абстрактное обозначение задачи, решаемой некоторой алгоритмической системой (общей проблемы в терминологии [8]);

D_z — допустимая конкретная проблема — конкретная реализация общей проблемы Z, таким образом, Z отождествляется с множеством D_Z допустимых конкретных проблем: $Z \Leftrightarrow D_Z{=}\{D_z\}$;

A — абстрактное обозначение некоторого алгоритма (финитного 1-процесса, доставляющего решение общей проблемы [8]), решающего задачу (общую проблему) Z;

$A_S = \{A_k | k = \overline{1,m}\}$ — алгоритмическая система, объединяющая m алгоритмов A_k, решающих задачу Z;

D — вход алгоритма A — конечное множество слов фиксированной длины в бинарном алфавите, задающее конкретную проблему D_z для общей проблемы Z;

R — результат (правильный ответ для конкретной проблемы), получаемый после обработки входа D моделью вычислений по алгоритму A, таким образом, алгоритм задает отображение: $D \underset{A}{\rightarrow} R$;

R_t — множество промежуточных результатов, порождаемых моделью вычислений при обработке входа D по алгоритму A; если R_j — множество промежуточных результатов после выполнения моделью вычислений j-й базовой операции, то

$$R_t = \{R_j | j = \overline{1, f_A(D)}\};$$

$D_A = \{D | D \underset{A}{\rightarrow} R\}$ — множество допустимых входов алгоритма A на задаче Z, в общем случае множества D_A и D_Z совпадают — $D_A = D_Z$;

$\lambda(D)$ — длина входа алгоритма: $D \underset{\lambda}{\rightarrow} N$, целочисленная функция, в общем случае определяемая как мощность множества D: $\lambda(D) = |D|$, в частных случаях (например, для задачи умножения матриц) — некоторая функция мощности: $\lambda(D) = g(|D|)$;

$f_A(D)$ — трудоемкость алгоритма A на входе D, целочисленная функция — число заданных алгоритмом A базовых операций принятой модели вычислений на входе D;

$V_A(D)$ — функция объема памяти, целочисленная функция — максимальное число задействованных алгоритмом A на входе D ячеек информационного носителя модели вычислений;

$\Psi_A(D)$ — функция ресурсной эффективности алгоритма A на входе D, может представлять собой, например, взвешенную сумму трудоемкости алгоритма и его емкостной эффективности на данном входе;

D_n — множество допустимых входов алгоритма A, имеющих длину n:

$$D_n = \{D | D \in D_A, \lambda(D) = n\};$$

$f_A^{\wedge}(n)$ — трудоемкость алгоритма в худшем случае на всех допустимых входах длины n, т. е. максимум $f_A(D)$ на множестве D_n; аналогично определяются $V_A^{\wedge}(n)$ и $\Psi_A^{\wedge}(n)$.

2. Кластеризация и распознавание входов алгоритма

Основная идея авторов статьи состоит в сведении различных методов построения алгоритмических систем к особенностям результатов решения задач кластеризации и распознавания входов, т. е. (более детально) в использовании результатов кластеризации множества известных входов $D_h \subset D_A$ алгоритмической системы A_S в некотором метрическом пространстве параметров с целью создания классов входов — результата процедуры кластеризации, с последующим определением рациональных алгоритмов обслуживания выделенных классов. Такие классы являются основой для решения задачи распознавания текущего входа алгоритмической системы, а полученное решение — принадлежность входа к определенному классу — может быть использовано для определения рационального, в смысле ресурсной эффективности, алгоритма системы или для построения некоторой рациональной комбинации алгоритмов для обработки текущего входа. В определенных случаях объектами кластеризации могут быть и множества промежуточных данных R_t, задаваемых некоторым алгоритмом системы A_k, на текущем входе. Наша цель — показать, что этот подход является конструктивным в аспекте создания классификации методов построения алгоритмических систем.

Для исключения возможных разногласий внесем некоторые уточнения в используемую терминологию. Достаточно часто между понятиями “классификация”, “кластеризация” и “распознавание” не делают особых различий. В целом терминология кластерного анализа достаточно разнообразна [9–12]. Первичным является *процесс выявления*

структуры на определенном множестве объектов, в частности — выделение подмножеств объектов, имеющих определенную общность. Для обозначения данного процесса используются термины: группировка, кластеризация, таксономия, классифицирование и классификация. *Результатом* этого процесса являются классы объектов (классификация), называемые также кластерами. Следующий процесс — *процесс определения принадлежности* нового объекта одному из уже существующих классов, обозначается терминами: распознавание, идентификация, узнавание. Таким образом, иногда термином классификация обозначается как процесс выявления структуры, так и результат данного процесса. В связи с этим авторы будут использовать следующую терминологию.

1. *Кластеризация* — процесс выявления структуры, т.е. разбиение какого-либо множества объектов на подмножества по определенному критерию. Другими словами, это процесс образования классов, в каждом из которых все объекты "близки" друг другу в определенном смысле, а именно, по значению используемой метрики пространства кластеризации.

2. *Классификация* — результат процесса кластеризации объектов. Классы — это выделенные подмножества исходного множества объектов, а совокупность выявленных классов есть классификация этих объектов. Заметим, что сами классы могут быть объединены далее в структурную иерархию. Следующий уровень иерархии классов — совокупность классов, обладающих определенной общностью, будем в дальнейшем называть *типом*.

3. *Распознавание* — процесс соотнесения нового исследуемого объекта с одним из уже полученных классов, рассматриваемых как результат процесса предшествующей кластеризации, тем самым распознавание возможно только при наличии уже выделенных классов. Такую постановку будем называть *задачей статического распознавания*. Заметим, что в рамках более общей постановки задачи распознавания, когда границы первичных классов в пространстве параметров уже известны, не исключается возникновение новых классов в процессе решения задачи распознавания объектов — такую постановку будем называть *задачей динамического распознавания*. Несмотря на возможность возникновения новых классов, первичное задание классов в этом процессе предполагается априорно в отличие от процесса кластеризации, где задача разбиения множества объектов решается "с чистого листа".

Рассмотрим кратко наиболее широко применяемые алгоритмы кластеризации. Для решения задачи кластеризации в настоящее время предложены разнообразные алгоритмы [10–13], и в принципе для кластеризации известных входов алгоритмической системы может быть использован любой из них. Выделим три алгоритма

1. Алгоритм кратчайшего незамкнутого пути (КНП), или "Краб" [10, 13]. Такой путь также называют минимальным покрывающим деревом, каркасом, или остовным деревом [11] взвешенного графа. Суть метода состоит в том, что две ближайшие точки, представляющие объекты в некотором метрическом пространстве, соединяются ребром, вес которого (длина) равен расстоянию между этими точками в выбранной метрике. Далее среди оставшихся отыскивается точка, ближайшая к одной из уже соединенных точек, и присоединяется к ним ребром, процесс продолжается до исчерпания всех точек — в результате получаем остовное дерево. Р. Прим доказал [12], что построенный таким способом граф имеет минимальную общую длину ребер среди всевозможных соединений, связывающих все вершины. Число необходимых классов k задается исследователем, и в найденном КНП (остовном дереве) отбрасываются $k-1$ дуг, дерево распадается на k деревьев, вершины которых образуют k классов. При этом не всегда отбрасываются

самые длинные дуги. Удаление дуг происходит по принципу максимизации выбранного критерия качества [13]. Данный метод позволяет выделять классы произвольной формы.

2. Алгоритм "Форель". На практике алгоритм КНП чаще всего используют в комбинации с алгоритмом "Форель", в котором случайный объект объявляется центром класса; все объекты, находящиеся от него на расстоянии не больше R (по выбранной метрике пространства), входят в первый класс. В этом классе определяется центр масс, который объявляется новым центром класса, процесс продолжается до стабилизации центра. Затем все объекты, попавшие в первый класс, изымаются, и процедура повторяется с новым случайным центром.

3. Комбинация алгоритмов "Форель" и КНП. Идея комбинирования двух алгоритмов состоит в том, что для заданного числа классов k при небольшом радиусе r по алгоритму "Форель" находят $k' > k$ классов, их центры соединяют методом КНП, из которого удаляют $k - 1$ ребер по принципу максимизации выбранного критерия качества [13] и получают k результирующих классов. При этом образуются классы более сложной формы, чем m-мерные шары. Здесь важна сама идея двухэтапной кластеризации: сначала выделяются заведомо компактные маленькие группы одним алгоритмом, а затем производится их объединение, используя другой алгоритм. Такая комбинация позволяет успешно выделять классы из довольно больших множеств, включающих сотни и тысячи объектов.

Использование описанной комбинации алгоритмов кластеризации позволяет провести кластеризацию накопленных входов решаемой задачи — входов алгоритма, и тем самым построить первичные классы входов. Тогда основной задачей создания алгоритмической системы является задача установления соответствия между выделенными классами входных данных (и, может быть, динамически добавленными классами) и входящими в алгоритмическую систему алгоритмами решения исследуемой задачи. Такое соответствие может быть установлено заранее (статически) на основе результатов теоретического анализа алгоритмов, выбранных для создания системы, или динамически, по мере обнаружения устойчивых классов входных данных и экспериментального определения наиболее рационального алгоритма для выделенных классов входов.

Поскольку предметом исследования в настоящей работе являются методы построения алгоритмических систем, то эти способы установления соответствия между алгоритмами и классами входных данных и будут использоваться в дальнейшем с учетом принятой терминологии для создания классификации самих методов построения алгоритмических систем.

3. Типы алгоритмических систем

Напомним, что принципиальная возможность создания алгоритмических систем обусловлена достаточным алгоритмическим разнообразием для широкого круга практически важных задач. В рамках такого пути совершенствования алгоритмического обеспечения можно выделить два типа алгоритмических систем, существенно отличающихся друг от друга, которые авторы предлагают различать и с терминологической точки зрения.

I. *Алгоритмический комплекс.* Под термином *алгоритмический комплекс* будем понимать совокупность различных алгоритмов решения данной задачи, организованную

таким образом, что *только один* из них выбирается как наиболее рациональный для текущего входа. Такой выбор может быть основан:

— на результатах теоретического ресурсного анализа каждого алгоритма из рассматриваемой совокупности (рис. 1, 2), таким образом, классы входов определяются на основе предварительного исследования алгоритмов. Выбор рационального алгоритма осуществляется в результате решения задачи статического распознавания принадлежности текущего входа к одному из заранее выделенных классов входов с априорно назначенными для этих классов рациональными алгоритмами;

— на динамическом обучении алгоритмического комплекса (рис. 3, 4), т. е. решении задачи динамического распознавания. Выбор рационального алгоритма осуществляется по результатам решения задачи кластеризации входов с последующим исследованием полученных классов и назначением рационального алгоритма для каждого класса входных данных.

В общем случае управляющий модуль алгоритмического комплекса на основе анализа параметров текущего входа, например такого параметра как размерность решаемой задачи (длина входа), осуществляет выбор алгоритма, рационального по заранее определенному критерию качества (см., например, [14]). Еще раз подчеркнем, что в алгоритмическом комплексе задача для конкретного входа решается только одним алгоритмом.

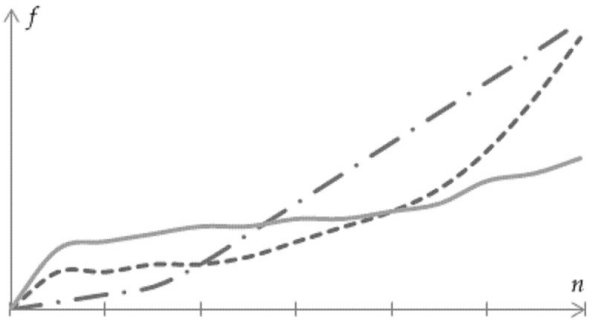

Рис. 1. Теоретический ресурсный анализ алгоритмов, входящих в комплекс; $f(\mathrm{A1})$ — штрихпунктир, $f(\mathrm{A2})$ — штрих, $f(\mathrm{A3})$ — сплошная линия

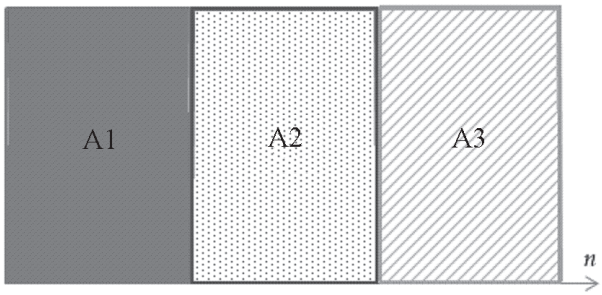

Рис. 2. Выявление классов и назначение алгоритмов на основе результатов ресурсного анализа; A1 — класс 1, A2 — класс 2, A3 — класс 3

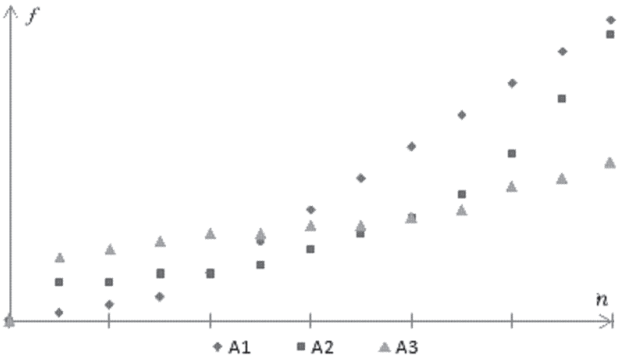

Рис. 3. Результаты динамического обучения алгоритмического комплекса

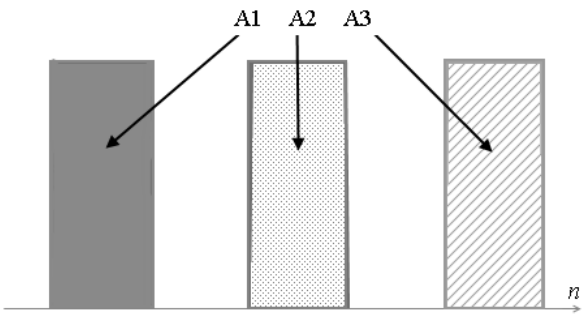

Рис. 4. Кластеризация входов и назначение рационального алгоритма каждому классу

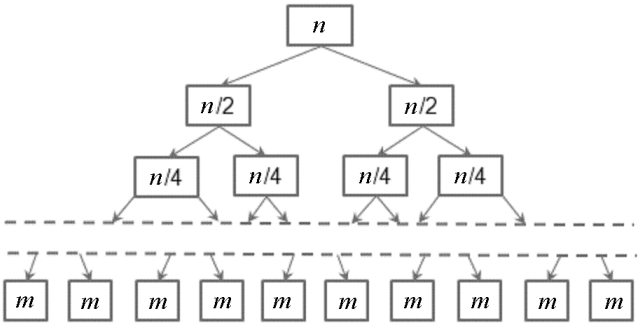

Рис. 5. Схема комбинации рекурсивного и итерационного алгоритмов; m — размерность промежуточных данных, при которой происходит останов рекурсии

II. *Комбинированный алгоритм.* Под термином *комбинированный алгоритм* будем понимать совместное использование (комбинацию) нескольких алгоритмов с различными ресурсными характеристиками при решении задачи для конкретного входа. Алгоритмы, входящие в комбинированный алгоритм, являются рациональными по выбранному критерию оценки качества для различных сегментов размерности задачи или других параметров входа. Такой путь, очевидно, приемлем, если, например, в основу решения исследуемой задачи положен метод декомпозиции, который предполагает последовательное понижение размерности задачи и, следовательно, возможность выбора для текущей размерности наиболее рационального алгоритма из имеющейся совокупности. Основные вопросы, возникающие при создания таких комбинированных алгоритмов, это вопрос о выборе комбинируемых алгоритмов и вопрос о выборе рационального порога переключения между ними. Ответы на них могут быть также получены на основе результатов решения задачи кластеризации входов — определения классов входов, и последующего решения задачи статического или динамического распознавания. Характерный и достаточно общий пример создания ресурсно-эффективных комбинированных алгоритмов — комбинирование рекурсивных алгоритмов с итерационными (рис. 5) [15]. При этом за счет сокращения порожденного дерева рекурсии путем замены его нижних уровней вычислениями, которые выполняются итерационным алгоритмом, более эффективным для текущей длины входа, в ряде случаев может быть достигнуто значительное улучшение совокупной ресурсной характеристики комбинированного алгоритма [2].

4. Классификация алгоритмических комплексов

Выбор рациональных по ресурсной эффективности алгоритмов, входящих в алгоритмический комплекс, для разных входов может происходить как априорно, так и непосредственно во время работы алгоритмического комплекса. Исходя из этого в типе алгоритмических комплексов мы выделяем два класса — алгоритмические комплексы без памяти и комплексы с памятью, или обучаемые алгоритмические комплексы.

Класс *алгоритмических комплексов без памяти* характеризуется тем, что выбор одного из алгоритмов комплекса происходит на основе априорных знаний — на основе теоретического исследования алгоритмов комплекса выявляются классы входов, в каждом из которых определенный алгоритм является наиболее эффективным — это теоретическое решение задачи кластеризации входов. При запуске алгоритмического

комплекса происходит распознавание входа — определение его принадлежности к одному из теоретически выделенных классов, после чего вход обрабатывается с помощью алгоритма, заранее предписанного данному классу входов. Тем самым алгоритмические комплексы данного класса порождаются решением задачи статического распознавания текущего входа алгоритма, при этом классы входов находятся по результатам теоретического анализа исследуемой совокупности алгоритмов. Как правило, пространством кластеризации в таком случае является множество значений актуальных длин входа рассматриваемой задачи.

В понимании оптимальности алгоритма как алгоритма, обладающего максимальной временной эффективностью, теоретический анализ предполагает получение функций трудоемкости исследуемых алгоритмов, аргументом которых является длина входа (размерность задачи) или некоторая мера этой длины. Переход к программным реализациям приводит к рассмотрению функций времени выполнения, получаемых из функций трудоемкости с использованием экспериментально определяемого среднего времени выполнения базовых операций [2]. Последующий совместный анализ позволяет указать наиболее рациональные алгоритмы для определенных сегментов длин входов рассматриваемой задачи. Таким образом, задача распознавания входа вырождается в задачу определения сегмента длин входа, в который попадает длина текущего входа данного алгоритмического комплекса.

В более общей постановке речь идет о теоретическом построении пространства параметров входов, связанных с конкретной задачей, решаемой алгоритмическим комплексом, разбиении этого пространства на непересекающиеся области — классы входов, и назначении для данных областей ресурсно-эффективных алгоритмов. Такое разбиение происходит на основе теоретического ресурсного анализа алгоритмов, а назначение рационального алгоритма текущему входу — на основе решения задачи распознавания принадлежности входа к одной из областей в пространстве параметров.

Если алгоритмический комплекс без памяти включает в себя два и более алгоритма, то мы имеем дело с *алгоритмическим комплексом без памяти с выбором*. Примером может служить комплекс решения задачи одномерной оптимальной по стоимости упаковки, включающий рекурсивный и табличный алгоритмы решения этой задачи [14].

Тривиальное вырождение алгоритмического комплекса без памяти связано с тем, что при всех возможных входах только один алгоритм является наиболее ресурсно-эффективным. Можно условно говорить об “алгоритмическом комплексе без памяти, без выбора”. Так, например, при нахождении наибольшего общего делителя для любого входа единственным эффективным алгоритмом является алгоритм Евклида [3].

Перейдем к рассмотрению *алгоритмических комплексов с памятью*. В этом подходе предполагается, что до момента запуска такого комплекса отсутствуют данные об областях эффективности входящих в него алгоритмов. Поэтому первый этап работы комплекса — обучающий цикл с последующей кластеризацией входов.

Этап обучения при решении каждой конкретной задачи может быть различным, но в общем случае он будет выглядеть следующим образом (см. рис. 3). На вход алгоритмического комплекса последовательно подается некоторый массив различных входов, который обрабатывается всеми входящими в него алгоритмами. Значения оценок ресурсной эффективности измеряются и запоминаются для каждого обработанного входа. На базе накопленных знаний производится кластеризация входов в пространстве параметров, и при условии выделения устойчивых классов происходит назначение каждому классу своего эффективного (по экспериментальным данным) алгоритма. Полученные

классы далее рассматриваются как предопределенные классы входов, для которых установлено однозначное соответствие с множеством алгоритмов комплекса.

За этапом обучения начинается этап эксплуатации — распознавание текущего входа и решение конкретной задачи, определенной данным входом, алгоритмом, предписанным данному классу. Если очередной вход невозможно отнести ни к одному из ранее полученных классов (задача динамического распознавания), то следует вернуться к этапу обучения, в конце которого должно быть принято решение — расширить один из уже известных классов или выделить новый класс со своим эффективным алгоритмом. После этого алгоритмический комплекс продолжает работать в режиме эксплуатации с обновленными знаниями.

Примером алгоритмического комплекса с памятью может служить самонастраивающийся (самосовершенствующийся) алгоритм сортировки, предложенный в [6].

5. Классификация комбинированных алгоритмов

В общем случае при создании комбинированных алгоритмов рассматривается ситуация, когда в комбинации участвуют несколько алгоритмов, хотя, как правило, обычно рассматривается два. При разработке комбинированных алгоритмов достаточно важным является вопрос об определении порога переключения с одного алгоритма, участвующего в комбинации, на другой. Существующие разнообразные подходы к решению этой задачи могут быть взяты в основу выделения классов комбинированных алгоритмов.

Рассмотрим вариант комбинирования двух алгоритмов — A1 и A2, оставляя обозначение A для результирующего алгоритма. Между алгоритмами, входящими в комбинацию, существует фиксированный порог переключения для каждого конкретного входа. Этот порог является границей между классами, на которые в ходе кластеризации по признаку рационального алгоритма разбивается вся область промежуточных значений. Переключение с одного алгоритма на другой производится при достижении некоторого порога переключения μ. Содержательно этот порог может быть, например, значением размерности подзадачи, если комбинированный алгоритм разрабатывается методом декомпозиции, или некоторым характерным параметром множества промежуточных результатов $\mu = \mu(R_t)$, на основе которого один из алгоритмов может быть выбран как более предпочтительный в текущей ситуации обработки входа в смысле выбранной функции качества $\Psi_{\mathrm{A}}(D)$. Тогда

$$\Psi_{\mathrm{A}}(D, \mu) = \Psi_{\mathrm{A1}}(D, \mu) + \Psi_{\mathrm{A2}}(D, \mu).$$

Полагая, что выбор оптимального порога переключения минимизирует совокупные ресурсные затраты $\Psi_{\mathrm{A}}(D, \mu)$, имеем

$$\mu^* = \arg\min_{\mu} \Psi_{\mathrm{A}}(D, \mu),$$

и принцип работы комбинированного алгоритма может быть описан следующим образом. После очередной итерации алгоритма A1 вычисляется значение параметра μ и сравнивается с пороговым, т. е. производится распознавание промежуточных результатов, на основе которого либо выполняется следующая итерация алгоритма A1, либо управление передается алгоритму A2.

На этой основе авторы предлагают различать три следующих, связанных со способом определения порога переключения μ^* подхода, определяющих классы комбинированных алгоритмов:

— стационарный подход, при котором в процессе разработки комбинированного алгоритма при теоретической оптимизации функции качества получаемый оптимальный порог переключения алгоритмов не зависит от особенностей входа и является стационарно оптимальным для всего диапазона входов, определяемых областью применения:

$$\mu \neq \mu(D), \quad \mu^* = \text{const}.$$

Использование этого подхода порождает *класс стационарно-адаптивных к входу комбинированных алгоритмов* с порогом переключения, фиксированным в момент разработки комбинированного алгоритма;

— статически-адаптивный подход, при котором получаемый в процессе разработки оптимальный порог переключения алгоритмов зависит от особенностей входа. При этом оптимальный порог (пороги) переключения рассчитывается управляющим модулем для текущего входа непосредственно перед запуском собственно комбинированного алгоритма, т. е. выполняется статическая адаптация к текущему входу алгоритма:

$$\mu = \mu(D).$$

Данный подход порождает *класс статически адаптивных к входу комбинированных алгоритмов* с вычислением порога переключения в момент запуска алгоритма;

— динамически-адаптивный подход, при котором оптимальный момент переключения между комбинируемыми алгоритмами определяется в динамике их выполнения на текущем входе на основе вычисления значения специальной управляющей функции. Для этой функции либо пороговое значение (простая динамическая адаптация к входу) известно теоретически, либо оптимальный порог и момент переключения определяются на основе совокупного анализа как текущего входа, так и определенной информации о ряде предыдущих входов (динамическая адаптация с памятью входов):

$$\mu = \mu(D, R_t).$$

В итоге получаем *класс динамически-адаптивных к входу комбинированных алгоритмов* с вычислением пороговой функции в динамике выполнения.

В качестве примера стационарно-адаптивного комбинированного алгоритма приведем алгоритм сортировки слиянием с сокращенным деревом рекурсии, в котором при длине текущего массива, меньшей или равной 12, происходит останов рекурсии и дальнейшая сортировка производится методом прямого включения [15].

Два других класса комбинированных алгоритмов могут быть проиллюстрированы на примере задачи одномерной оптимальной по стоимости упаковки (задача о рюкзаке). В силу аддитивности целевой функции метод динамического программирования применим для решения этой задачи и позволяет получить ее точное решение. Два известных алгоритма, реализующих этот метод, — табличный и рекурсивный, имеют различные ресурсные характеристики, что создает возможность разработки комбинированного алгоритма, в котором оптимальный порог переключения между алгоритмами зависит от особенностей входа. Расчет значения этого порога при условии, что оценкой качества алгоритмов является временная эффективность, может быть выполнен с использованием

теоретических функций трудоемкости комбинируемых алгоритмов, при этом значением порога является номер типа груза.

Статически-адаптивный комбинированный алгоритм будет содержать управляющий модуль, который для конкретного входа рассчитывает порог переключения и запускает табличный алгоритм для определенной пороговым значением части типов грузов. Второй фазой запускается рекурсивный алгоритм, который в листьях порожденного дерева рекурсии копирует результаты табличного алгоритма и цепочкой рекурсивных возвратов получает оптимальное решение для данного входа. Детально этот комбинированный алгоритм описан в [14].

Динамически-адаптивный подход реализован в волновом алгоритме [16]. Он не производит предвычисления порога переключения. Вместо этого после каждого прохода рекурсивного алгоритма подсчитывается число заполненных строк структуры данных. Если число этих строк становится больше порога, в данном случае равного половине общего числа строк структуры данных, то управление передается табличному алгоритму.

6. Классификация методов построения алгоритмических систем

Анализ приведенных данных показал, что корнем иерархической классификации являются алгоритмические системы, т. е. алгоритмы, объединенные разнообразными связями, причем, если эти связи существуют только с управляющим модулем системы (система представляет собой дерево глубины два), то мы получаем алгоритмический комплекс, если же они существуют еще и между самими алгоритмами (и, следовательно, алгоритмическая система имеет более сложную структуру), то имеем комбинированный алгоритм.

Таким образом, предлагается следующая *классификация методов построения алгоритмических систем* (рис. 6), согласно которой в алгоритмических системах выделяются два крупных типа — алгоритмические комплексы и комбинированные алгоритмы. Алгоритмические комплексы включает в себя два класса — обучаемые алгоритмические комплексы (комплексы с памятью) и алгоритмические комплексы без памяти.

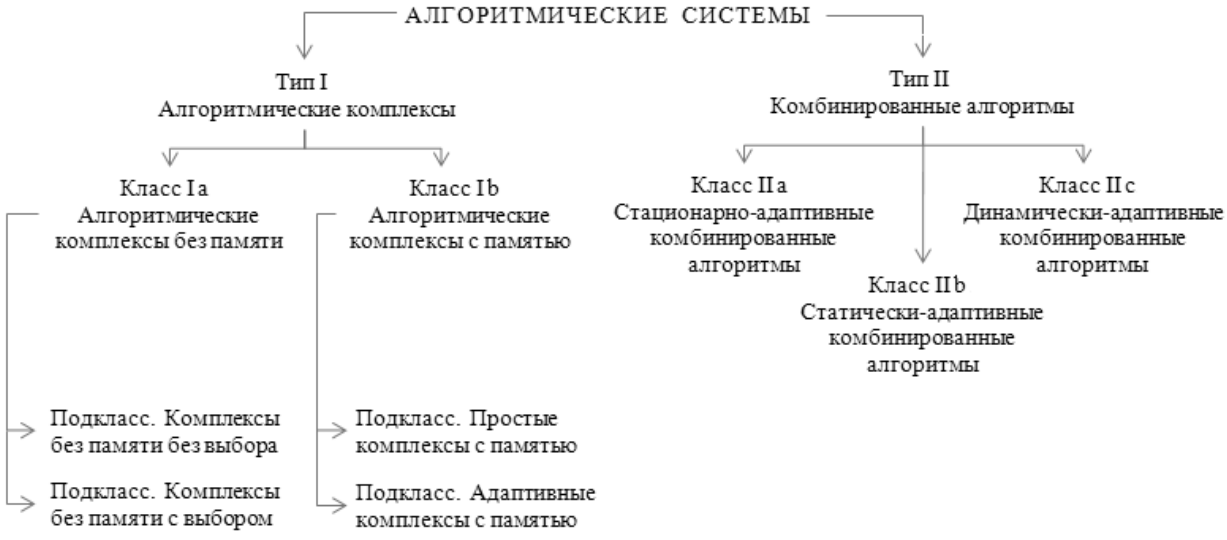

Рис. 6. Классификация алгоритмических систем (схема)

Первые представлены двумя подклассами — простые комплексы с памятью и адаптивные комплексы с памятью. Последние, в свою очередь, также представлены двумя подклассами — алгоритмические комплексы без памяти, без выбора и комплексы без памяти, с выбором. Во втором типе выделяются три класса — стационарно-адаптивные, статически-адаптивные и динамически-адаптивные комбинированные алгоритмы.

Ниже приведено описание связи предложенной классификации методов построения алгоритмических систем со способами классификации и распознавания входов и методами определения рациональных алгоритмов.

Алгоритмические системы

Алгоритмическая система $A_S = \{A_k | k = \overline{1,m}\}$ представляет собой определенным образом организованное объединение m различных алгоритмов A_k решения задачи (общей проблемы) Z.

Тип I. Алгоритмические комплексы

Алгоритмическая система $A_S = \{A_k | k = \overline{1,m}\}$ решения задачи Z, организованная как набор различных алгоритмов таким образом, что управляющий модуль комплекса выбирает для решения конкретной проблемы, заданной входом D_z, только один из алгоритмов комплекса — алгоритм A_k как наиболее рациональный по ресурсным характеристикам.

Класс Ia. Алгоритмические комплексы без памяти включают следующие подклассы:

— *описание пространства кластеризации входов* на основе теоретической параметризации входов, учитывающей особенности ресурсных характеристик алгоритмов комплекса;

— *определение классов входов* на основе теоретического разграничения областей ресурсной эффективности для алгоритмов комплекса в пространстве параметризации;

— *назначение рациональных алгоритмов для полученных классов*, определяемое на основе теоретического анализа ресурсной эффективности алгоритмов комплекса в классах;

— *выбор алгоритма для текущего входа* по результатам решения задачи статического распознавания текущего входа управляющим модулем комплекса.

Подкласс алгоритмических комплексов без памяти без выбора является тривиальным вырождением комплекса до одного алгоритма.

Класс Ib. Алгоритмические комплексы с памятью включают следующие подклассы:

— *описание пространства кластеризации входов* на основе теоретической параметризации входов, учитывающей особенности ресурсных характеристик алгоритмов комплекса;

— *определение классов входов* первично по результатам этапа обучения, состоящего в определении классов на основе кластеризации известной выборки типичных входов с возможным вторичным добавлением классов в режиме эксплуатации — при этом управляющий модуль решает задачу динамического распознавания;

— *назначение рациональных алгоритмов для полученных классов*, определяемое на основе экспериментального анализа ресурсной эффективности алгоритмов комплекса на этапе обучения и в момент образования нового класса входов;

— *выбор алгоритма для текущего входа* по результатам динамического распознавания текущего входа управляющим модулем комплекса.

Подкласс адаптивных комплексов с памятью отличается тем, что при создании нового класса входов в режиме эксплуатации происходит адаптивная модификация одного из алгоритмов комплекса для наиболее эффективной обработки входов из нового класса, т. е. управляющий модуль выполняет адаптацию комплекса к новому классу входов.

Тип II. Комбинированные алгоритмы

Алгоритмическая система $A_S = \{A_k | k = \overline{1,m}\}$ решения задачи (общей проблемы) Z, организованная таким образом, что при решении конкретной проблемы, заданной входом D_z, используется комбинация нескольких алгоритмов системы с соответствующими этому входу настройками, обеспечивающая наилучшие ресурсные характеристики системы для этого входа.

Класс IIa. Стационарно-адаптивные комбинированные алгоритмы с порогом переключения, фиксированным в момент разработки комбинированного алгоритма, — классы входов выделены на этапе разработки комбинированного алгоритма, задача распознавания решена априорно для всех входов.

Класс IIb. Статически-адаптивные комбинированные алгоритмы с вычислением порога переключения в момент запуска алгоритма — решается задача распознавания входа как задача статического распознавания принадлежности к классам входов, полученным по результатам теоретического анализа комбинированных алгоритмов.

Класс IIc. Динамически-адаптивные комбинированные алгоритмы с вычислением пороговой функции в динамике выполнения — решается задача статического распознавания принадлежности множества промежуточных результатов алгоритма к классам, определенным в результате теоретического анализа комбинированных алгоритмов.

Таким образом, в статье предложена оригинальная классификация алгоритмических систем, в основу которой положены возможные результаты кластеризации и последующего распознавания входов и промежуточных данных алгоритма. Представленную классификацию можно рассматривать как базу для создания методик разработки алгоритмических систем, учитывающую особенности их построения и повышающую ресурсную эффективность алгоритмических систем, так как при ее использовании обеспечивается связь назначения рациональных алгоритмов с устойчивыми классами входов с обработкой текущего входа алгоритмом, определенным на основе решения задачи распознавания.

Список литературы

[1] Ульянов М.В. Классификация и методы сравнительного анализа вычислительных алгоритмов. М.: Физматлит, 2004. 212 с.

[2] Ульянов М.В. Ресурсно-эффективные компьютерные алгоритмы. Разработка и анализ. М.: Физматлит, 2008. 304 с.

[3] Кормен Т., Лейзерсон Ч., Ривест Р., Штайниз К. Алгоритмы: Построение и анализ. 2-е изд.: Пер. с англ. М.: Издательский дом "Вильямс", 2005. 1296 с.

[4] Ахо А., Хопкрофт Дж., Ульман Дж. Структуры данных и алгоритмы: Пер. с англ. М.: Издательский дом "Вильямс", 2001. 384 с.

[5] Чмора А.Л. Современная прикладная криптография. М.: Гелиос АРВ, 2001. 256 с.

[6] Ailon N., Chazelle B., Comandur S., Liu D. Self-improving algorithms // SODA '06: Proc. of the Seventeenth Annual ACMSIAM Symp. on Discrete Algorithm. New York, USA: ACM, 2006. P. 261–270.

[7] Грешилов А.А. Прикладные задачи математического программирования. М.: Логос, 2006. 288 с.

[8] Post E.L. Finite combinatory process — formulation 1 // J. Symbolic Logic. 1936. No. 1. P. 103–105.

[9] Олдендерфер М.С., Блэшфилд Р.К. Кластерный анализ. Факторный, дискриминантный и кластерный анализ: Пер. с англ. / Под. ред. И.С. Енюкова. М.: Финансы и статистика, 1989. 215 с.

[10] Лагутин М.Б. Наглядная математическая статистика: Уч. пособие. 2-е изд. М.: БИНОМ. Лаборатория знаний, 2009. 472 с.

[11] Свами М., Тхуласираман К. Графы, сети и алгоритмы. М.: Мир, 1984. 380 с.

[12] Prim R.C. Shortest connection networks and some generalization // Bell System Technical J. 1957. Vol. 36. P. 1389–1401.

[13] Загоруйко Н.Г. Прикладные методы анализа данных и знаний. Новосибирск: Ин-т математики СО РАН, 1999. 270 с.

[14] Гиряева А.Н., Наумова О.А., Ульянов М.В., Яковлев И.А. Комбинированный алгоритм решения задачи одномерной упаковки: статически-адаптивный подход // Проблемы полиграфии и издательского дела. 2008. № 2. С. 67–82.

[15] Гиряева А.Н., Исаева А.С., Наумова О.А. и др. Комбинированные итерационно-рекурсивные алгоритмы сортировки // Автоматизация и современные технологии. 2008. № 3. С. 17–26.

[16] Наумова О.А., Ульянов М.В. Комбинированный и волновой алгоритмы решения задачи упаковки: Принципы построения и особенности // Бизнес-информатика. 2009. № 2. С. 27–33.

Поступила в редакцию 25 февраля 2010 г.,
с доработки — 2 сентября 2010 г.