УДК 004.421

В.И. Шинкаренко

ФУНКЦИОНАЛЬНАЯ ЭФФЕКТИВНОСТЬ НЕЧЕТКО СПЕЦИФИЦИРОВАННЫХ АЛГОРИТМОВ

Определено понятие нечетко специфицированного алгоритма. Рассмотрены особенности спецификации таких алгоритмов. Выделена одна из их эксплуатационных характеристик – функциональная эффективность. Показана значимость этой характеристики для задач практического программирования. Предложены статистические показатели оценки функциональной эффективности.

Введение

Одной из задач прикладной теории алгоритмов является построение эффективных алгоритмов и их анализ [1, 2].

При этом под эффективностью алгоритмов, в первую очередь, понимают временную эффективность или, как чаще ее называют, временную сложность. Ставится задача оценки того, насколько эффективен алгоритм с точки зрения возможного времени его выполнения.

Интересна и другая задача: насколько эффективен алгоритм с точки зрения его функциональных “обязанностей”.

Начнем с того, что одним из основных свойств алгоритмов является их функциональность. Это свойство заключается в том, что алгоритм реализует некоторую, в математическом понимании, функцию

$$y = f(x), \qquad (1)$$

где $x \in X$, $y \in Y$ - элементы некоторых произвольных множеств.

Данное свойство является одним из фундаментальных свойств алгоритмов наряду с дискретностью, результативностью, доступностью, массовостью, точностью (в смысле точной последовательности шагов), однозначностью [1].

Исследователи не заостряют внимания на этом свойстве, по-видимому, считая его тривиальным. Хотя понятие функциональной эквивалентности алгоритмов, как алгоритмов реализующих одинаковую функцию вида (1), является вполне устоявшимся.

В данной статье обращается внимание на это свойство ввиду его особенности применительно к нечетко специфицированным алгоритмам. Эта особенность заключается в том, что разработанный в соответствии с нечеткой спецификацией алгоритм может в каких-то случаях и в какой-то степени удовлетворять или нет требованиям пользователя.

Само понятие нечетко специфицированного алгоритма в некоторой мере алогично. Применительно к алгоритму спецификация определяется как “точное, однозначное, недвусмысленное описание” [3] алгоритма. Само определение подразумевает точность, четкость. Однако многие задачи из практики программирования, реализованные в алгоритмах и программах и принесшие значительную пользу, можно отнести именно к нечетко специфицированным.

Очевидно, что и понятие нечеткой спецификации алгоритма, и понятие функциональной эффективности требуют детальных пояснений и определений.

1. Нечетко специфицированные алгоритмы

Ввиду явного противоречия между точностью спецификаций и их нечеткостью понятие нечетко специфицированного алгоритма требует уточнения и как можно более строго определения.

Определение понятия нечетко специфицированного алгоритма дадим на основе анализа характерных примеров. Рассмотрим два класса алгоритмов: сжатия данных и кластеризации. При этом обратим внимание на те вопросы, которые будут нужны при последующем изложении: спецификация, разработка алгоритмов, особенности и оценка результатов их выполнения.

1.1. Алгоритмы сжатия данных. *Особенности спецификации.* Неформально алгоритмы сжатия данных можно специфицировать следующим образом:

спецификация 1. Алгоритм должен выполнять кодирование данных с возможностью последующего восстановления (декодирования), объем выходных данных должен быть как можно меньше.

Нечеткость формулировки заключается в требовании “как можно меньше”. Если бы ставилась задача минимизации объема закодированных данных, спецификация была бы четкой. Однако задача построения такого алгоритма является намного более сложной, чем первоначальная, а для случая сжатия без потери информации, в общем случае, - неразрешимой.

На основе этой нечеткой спецификации разработан целый ряд универсальных программ - архиваторов, таких, как RAR, ZIP и др.

Эффективность конкретного выполнения программы (КВП) оценивается показателем степенью сжатия данных.

Как известно [4], степень сжатия данных в значительной степени зависит от особенностей данных.

В первую очередь степень сжатия зависит от вида данных: текстовые, графические, аудио, видео, смешанные и т.д. Далее от характеристик представления данных. Например, для растровой графики: разрешения, глубины цвета, цветовой модели и т.п. И наконец, от самих данных. Так, результаты сжатия фотографии и одноцветного квадрата такого же размера будут сильно разниться.

С учетом особенности данных и условий применения алгоритмов сжатия спецификация конкретизируется. Например, спецификация дополняется следующими требованиями: кодированию подлежат растровые изображения, код должен быть устойчив к ошибкам и т.п. В соответствии с конкретизированными спецификациями разрабатываются более специализированные алгоритмы сжатия.

Особенности алгоритмов. Алгоритмы сжатия не могут не ориентироваться на структуру сжимаемых данных. Это следует из того факта [4], что без потери информации ни один алгоритм не обеспечит сжатия *любого* файла, т.е. любой алгоритм сжатия размеры одной части файлов уменьшит, а другой увеличит (либо в случае копирования не изменит).

Отметим три возможности учета структуры данных в алгоритмах сжатия:

- специализация алгоритмов для типичных структур (форматов) данных;
- адаптация алгоритмов к данным в процессе сжатия;
- параметрическое управление степенью сжатия.

Остановимся на последнем. Практически все алгоритмы сжатия имеют ряд параметров управления, значения которых в большей степени задаются в программах - архиваторах как константы и в силу сложной взаимосвязи их значений с результатом сжатия недоступны пользователю. Например, частью многих алгоритмов сжатия является алгоритм LPC (линейно – предсказывающее кодирование) [4]. Значение R-битного участка кода предсказывается как линейная комбинация нескольких предыдущих участков, и, если зависимости в исходных данных близки к линейным, выполнение алгоритма приводит к появлению многих повторяющихся участков кода. Управляющими здесь являются такие параметры, как длина участка кода и коэффициенты линейного прогноза.

Особенности результатов выполнения алгоритмов. Как оценить, какая степень сжатия информации является достаточной?

Рассмотрим пример. Одним из применений архиваторов является перенос данных с одной ЭВМ на другую с использованием некоторого носителя информации.

Допустим, необходимо перенести два файла, один объемом V_1 и второй - V_2. Посредником является носитель информации объемом V_0. При этом $V_1 >> V_0$ и $V_2 > V_0$. Для переноса воспользуемся архиватором, который позволяет сжать файлы до объемов $V_{1\,сж}$ и $V_{2\,сж}$ (рис. 1).

В первом случае, несмотря на высокую степень сжатия (2-3 раза), результат является неудовлетворительным, во втором же при низкой степени сжатия ($\approx 1{,}1$ раза) результат удовлетворительный.

Таким образом, какие-то результаты выполнения алгоритма будут приемлемы для пользователя, а какие-то нет. Опреде-

литься с этим можно лишь после выполнения программы, реализующей алгоритм.

Такая ситуация возникает как следствие нечеткости спецификации. Четкая спецификация, отвечающая свойствам ограниченности и обобщенности [5], предопределяет необходимые для пользователя результаты.

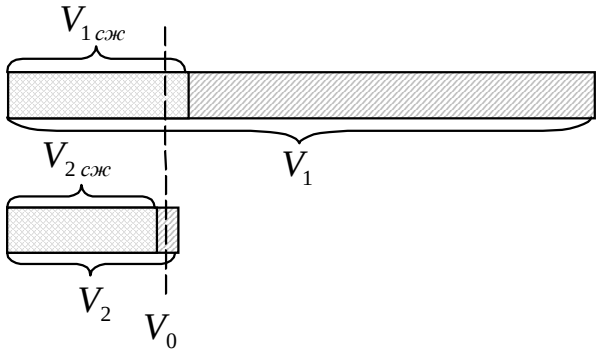

Рис.1. Пример результатов сжатия двух файлов

Сравнение алгоритмов. Практически алгоритмы сжатия сравниваются по их реализациям в программах-архиваторах.

Пример такого сравнения из [4] (частично) на основе файлов набора CalgCC в табл. 1.

Таблица 1. **Сравнение архиваторов по степени сжатия файлов**

Файлы	Архиваторы			
	ARJ	PKZIP	RAR	7-Zip
Bib	3.08	3.16	3.39	3.62
Bookl	2.41	2.46	2.80	2.94
Book2	2.90	2.95	3.39	3.59
…	…	…	…	…
Progp	4.32	4.37	4.57	4.73
Trans	4.65	4.79	5.23	5.56
Итого	3.47	3.55	3.80	4.10

Сравнивается степень сжатия отдельных файлов и средняя несколькими архиваторами (алгоритмами). При этом не определен порядок отбора, количество файлов (кстати, слишком небольшое), на которых выполняется сопоставление. Оценка может сильно измениться на другой выборке файлов.

1.2. Алгоритмы кластеризации. Согласно Б. Эверитту, “кластеры – это непрерывные области (некоторого) пространства с относительно более высокой плотностью точек, отделенные от других таких же областей областями с относительно низкой плотностью точек”[6].

Спецификация алгоритма кластеризации:

спецификация 2. Алгоритм должен выполнять анализ данных и выделение в них всех кластеров. Имеются в виду экспериментальные данные, данные, полученные из хранилищ, и т.п.

Нечеткость спецификации заключается в понятии “более и менее высокой плотности точек”. На рис. 2 приведены три примера кластеров.

Если в некоторых частных случаях (рис. 2,а) кластеры четко обозначены, то в более общем случае (рис. 2,б, 2,в) границы кластеров размыты. На рис. 2б, 2,в фактом является наличие двух кластеров, но ни одну точку нельзя с уверенностью причислить к кластеру либо межкластерному пространству, так как плотность точек постоянно снижается при удалении от центра кластера.

Можно с большой долей уверенности предположить, что построение универсального алгоритма кластеризации невозможно в силу значительного разнообразия исходных данных. Отметим лишь некоторые возможные их особенности:

- области с “относительно низкой плотностью точек” не содержат точек - нулевая плотность точек между кластерами (рис. 2,а);
- границы классов четко (рис. 2,а) либо нечетко (рис. 2,б, 2,в) выражены;
- кластеры образуют выпуклые области (рис 2,а, 2,б) либо области произвольной формы (рис. 2,в);
- точки однородные либо нет (с некоторыми атрибутами, например цветом), в случае с разнородными точками возможны пересекающиеся кластеры;
- иерархические кластеры - “кластеры в кластерах”;
- данные произвольных пространств с различной мерой близости.

Универсальный алгоритм кластеризации должен учитывать все возможные особенности данных, но лишь визуализация в двумерном пространстве заранее (до построения либо применения алгоритма) позволяет эти особенности определить. В общем случае N-мерного пространства эти

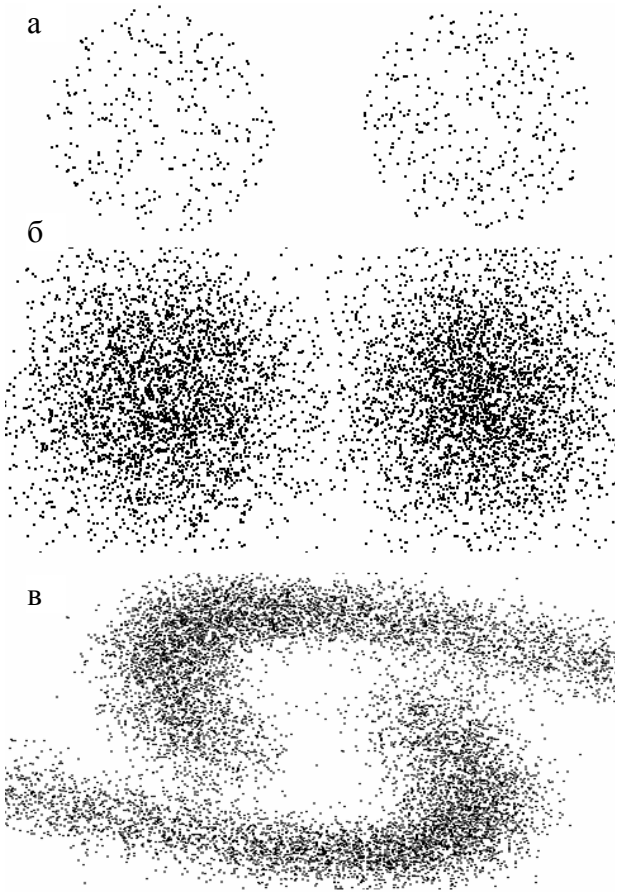

Рис.2. Кластеры в двумерном пространстве: а - точки равномерно распределены и четко разграничены; б, в – точки нормально распределены, границы кластеров размыты

особенности могут быть выявлены лишь в процессе самой кластеризации.

Разработанные эвристические [7] и математически обоснованные [6] алгоритмы могут с разной степенью эффективности применяться к различным исходным данным.

Для выявления других причин и последствий нечеткости спецификаций рассмотрим характерный алгоритм кластеризации - иерархический алгоритм [8].

Изначально все точки считаем центрами кластеров, т.е. имеем N кластеров по одной точке в каждом. Определяем минимальное расстояние между центрами кластеров. Такие два кластера объединяем в один, пересчитав координаты центра. Повторяем данную процедуру объединения кластеров до получения единственного.

В процессе объединения строим дерево (рис. 3). Каждый i-й его уровень представляет собой N-i корень дерева, который в свою очередь определяет все точки кластера.

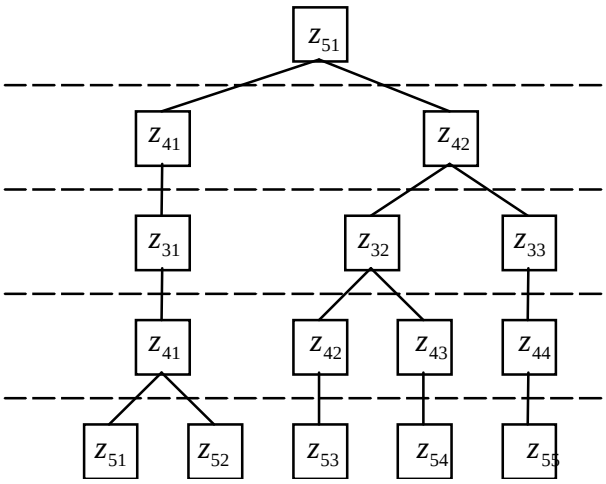

Рис. 3. Иерархия точек в процессе кластеризации.

Один из уровней иерархии и является результатом кластеризации. Какой именно? Для ответа на этот вопрос необходимо ввести некий критерий. В [16] указаны среднее квадратов расстояний между образами в кластере, среднее квадратов расстояний между образами, входящими в разные кластеры, показатели, основанные на понятии матрицы рассеивания, минимальная и максимальная дисперсии, сумма квадратов расстояний между точками и соответствующими центрами кластеров.

Выбор одного из них часто определяется практической целесообразностью в процессе оценки результатов кластеризации.

Таким образом, ***причины нечеткости спецификации*** алгоритмов кластеризации заключаются в следующем:

- заранее неизвестными особенностями данных, которые следует учесть в алгоритме;
- нечетким понятием “большей и меньшей плотности” точек;
- неоднозначностью показателей качества кластеров и результатов кластеризации.

При этом некоторые причины нечеткости являются неустранимыми на этапе спецификации.

Так как алгоритм нечетко специфицирован, то определить, удовлетворяют ли результаты выполнения алгоритма в каждом конкретном случае можно лишь в процессе валидации, т.е. на последней стадии це-

почки: спецификация →
алгоритм →
программа →
КВП →
валидация.

Если результаты неудовлетворительны, то следует возвращаться на несколько шагов назад в указанной цепочке, что повышает затраты на решение конкретных задач.

Какие меры применяются разработчиками алгоритмов для снижения затрат?

В случае с кластеризацией применяются различные управляющие данные для оперативного на этапе программы и КВП управления результатами. Управляющие параметры для некоторых алгоритмов кластеризации даны в табл. 2.

1.3. Общие особенности нечетко специфицированных алгоритмов. Анализ приведенных и других алгоритмов позволяет к закономерностям нечетко специфицированных алгоритмов отнести причины или источники нечеткости и средства устранения последствий нечеткости.

Основные источники нечеткости:

- заранее неизвестные особенности исходных данных;
- множественность показателей качества результатов;
- неопределенный уровень требований к результатам.

Устранение последствий нечеткости заключается в уточнении спецификаций, адаптации алгоритмов к данным и управлении алгоритмами посредством управляющих данных.

Уточнение спецификаций по отношению к данным дает возможность разрабатывать *новые* более эффективные, но и более специализированные алгоритмы.

Специализация алгоритмов, разработка адаптивных алгоритмов связаны с появлением *новых* алгоритмов, вопрос функциональной эффективности которых решается отдельно.

Наибольший интерес с точки зрения функциональной эффективности представляют алгоритмы с управлением.

Приведенные примеры нечетко специфицированных алгоритмов не являются какими-то исключительными.

К нечетко специфицированным можно отнести следующие алгоритмы: распознавания текста, речи и др. (в том числе нейросети), принятия решений, многокритериальной оптимизации, агенто-ориентированной технологии и др.

Как видим, нечеткие спецификации присущи многим, в основном эвристическим алгоритмам, которые относят к области искусственного интеллекта. С учетом бурного развития и перспектив этого направления нечетко специфицированные алгоритмы представляются как некоторая тенденция развития программирования.

1.4. Определение нечетко специфицированного алгоритма. Нечеткую спецификацию следует отличать от неполной.

Свойство полноты [3] или ограниченности [5] не выполняется, если в спецификации упущено какое-то требование. При этом возникает неоднозначность, в резуль-

Таблица 2. **Управляющие параметры алгоритмов кластеризации**

Алгоритм кластеризации	Управляющие параметры
Простой пороговый алгоритм [16]	Порог T
Максиминного расстояния [16]	Соотношение расстояний между вновь создаваемым и ранее выделенными кластерами
K внутригрупповых средних [16]	Количество кластеров
ИСОМАД [16]	5 параметров (ориентировочное количество кластеров, показатель компактности кластеров и др.)
Глобальный “Роден” [8]	Уровень качества α и уровень отделимости β
Локальный “Роден” [8]	Уровень качества α, уровень отделимости β и радиус кластеризации r

тате которой алгоритмы, удовлетворяющие спецификации, могут не устраивать пользователя.

Нечеткая спецификация также допускает неоднозначность. Принципиальным является то, что эту неоднозначность нельзя устранить на этапе спецификации. Результаты выполнения алгоритма не всегда будут удовлетворять пользователя.

Кардинальное отличие нечеткой спецификации от неполной в том, что нечеткость неустранима.

В случае четко специфицированных алгоритмов спецификацией задается одна или множество функций, необходимых пользователю для решения некоторой задачи и удовлетворяющих пользователя.

Если функция одна, то можно утверждать, что спецификация и алгоритм являются разными формами задания функции вида (1).

Возьмем, например, формализованную спецификацию алгоритма сортировки.

Спецификация 3. Предусловие

$$P \equiv \boldsymbol{X} = [x_1, x_2 \ldots x_N] : x_i \in Z,\ \boldsymbol{X} \in X, \quad (2)$$

где $\boldsymbol{X}$ – некоторый вектор с компонентами из некоторого линейно упорядоченного множества Z относительно отношения порядка $\propto$.

Постусловие

$$Q \equiv \boldsymbol{X}^* \in Y \equiv X : \boldsymbol{X}^* = [x_1^*, x_2^* \ldots x_N^*] : (\forall x_i^*\ n(x_i^*) = n(x_i)) \,\&\, \& (\forall i \in [1, N-1] \quad x_i^* \propto x_{i+1}^*), \quad (3)$$

где $n(x) = \sum_{i=1}^{N} \begin{cases} 1, \text{ если } x = x_i \\ 0, \text{ если } x \neq x_i \end{cases}$.

Эта спецификация определяет однозначное соответствие между элементами множества X и Y (рис. 4, а).

Спецификация многих алгоритмов задает неоднозначное отображение множества X в Y (рис. 4, б). Характерными представителями таких алгоритмов являются алгоритмы, реализующие методы приближенных вычислений. К примеру, спецификация алгоритма определения корня непрерывной функции одной переменной:

спецификация 4. $P \equiv (\varphi(p) \in C)\ \&$

$$(p \in [a,b])\ \&\ (\varphi(a) \cdot \varphi(b) < 0)\ \&\ (\varepsilon \in R),$$

где C – пространство непрерывных функций.

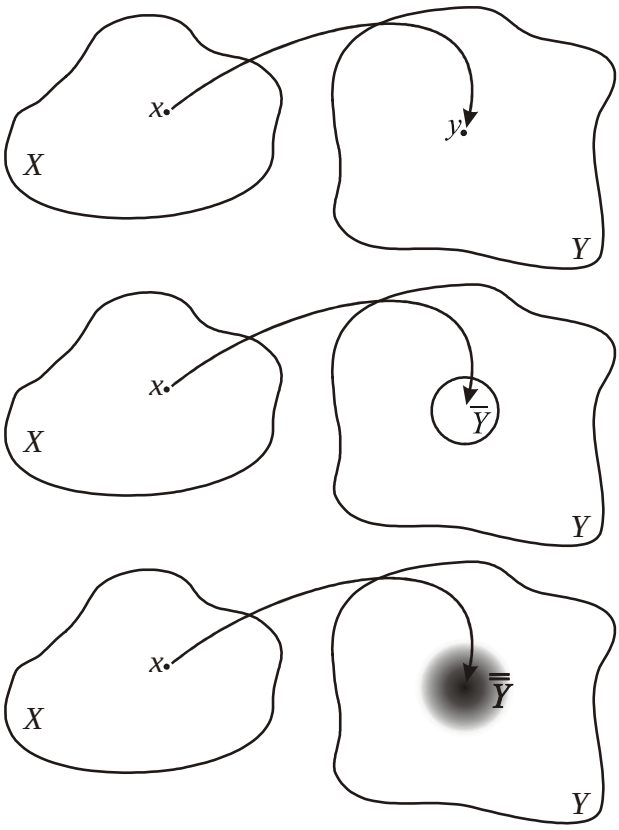

Рис.4. Отображение множества X на множество Y:
а) однозначное,
б) множественное в) нечеткое

$$Q \equiv p : (|p - p_0| < \varepsilon)\ \&\ (p_0 \in [a,b])\ \& \ \& (\varphi(p_0) = 0). \quad (4)$$

При этом (4) задает отображение элементов множества X $x = \{\varphi(p), a, b, \varepsilon\} \in X$ на элементы множества Y $y = \{p\} \in Y$. Заметим, что y может принимать разные значения при фиксированном x, а значит, спецификация задает множество функций вида (1). На рис. 4,б каждому значению $x \in X$ соответствует множество элементов $\overline{Y}$ из Y (в общем случае не обязательно ограниченное и связное). Каждая функция может быть реализована одним или несколькими алгоритмами. Так, методы хорд и касательных, соответствующие (4), реализуют различные функции, значения которых никогда не совпадают (только в пределе, который алгоритмически недостижим). Однако в этом случае любая функция из заданного множества и любой соответствующий ей алгоритм при КВП дают результаты, удовлетворяющие пользователя.

Следовательно, спецификация 4 является и полной и четкой.

Спецификации 1 и 2 также задают множественное отображение X в Y (рис. 4,в), но в отличие от предыдущего ни одно значение из $\overline{\overline{Y}}$ нельзя отнести к приемлемым. Можно говорить о степени принадлежности каждого элемента $y \in \overline{\overline{Y}}$ к удовлетворяющим пользователя результатам, т.е. множество $\overline{\overline{Y}}$ является нечетким. На рис.4. степень принадлежности y множеству удовлетворяющих пользователю результатов условно задана оттенками серого цвета (черный – степень принадлежности равна 1, белый – 0).

Определение 1. Спецификация алгоритма является нечеткой, если она задает отображение из X в Y, при этом каждому элементу из X ставится в соответствие нечеткое множество из Y.

2. Функциональная эффективность алгоритмов

Как отмечено выше, функция, реализуемая нечетко специфицированным алгоритмом, способна в *какой-то мере* удовлетворить потребностям пользователя. Встает вопрос, насколько эффективно такая функция им удовлетворяет?

Требуется дать объективный ответ на вопросы типа:

- насколько хорошо сжимает данные конкретный алгоритм;
- насколько хорошо выполняется кластеризация конкретным алгоритмом;
- какой алгоритм предпочтительнее с точки зрения функциональной эффективности.

Определение 2. Под функциональной эффективностью будем понимать эксплуатационную характеристику алгоритма, которая отражает степень соответствия алгоритма потребностям пользователя.

Любой четко специфицированный алгоритм реализует функцию, результаты которой всегда должны удовлетворять потребностям пользователя. Можно считать, что такой алгоритм имеет максимальную функциональную эффективность.

Задача оценки функциональной эффективности алгоритма, как, впрочем, и других эксплуатационных характеристик, возникает в двух случаях:

- при разработке новых, более совершенных алгоритмов для оценки их эффективности;
- при разработке прикладного программного обеспечения с целью выбора более эффективного алгоритма из альтернативных с учетом перспектив и особенностей применения разрабатываемого ПО.

3. Показатели функциональной эффективности алгоритмов

Показатель или показатели функциональной эффективности должны давать количественную оценку качества результатов выполнения алгоритма.

При этом оценка должна обобщенной.

В рассмотренных выше примерах подход к оценке функциональной эффективности различается.

В алгоритмах сжатия функциональная эффективность оценивается по степени сжатия, т.е. показатель зависит от двух измерений: исходных и результирующих данных.

В алгоритмах кластеризации имеем несколько показателей и все они основаны на измерении результирующих данных.

По-видимому, в общем случае следует опираться только на результирующие данные, так как измерения на исходных, как показывает второй пример, не всегда возможны.

Будем считать, что на множестве Y задана функция

$$\rho = \phi(y), \qquad (5)$$

где $\rho \in R^+$ - оценка качества полученных алгоритмом результатов.

Например, для алгоритмов сжатия данных $\rho = \phi(y)$ - функция, определяющая размер сжатых данных для конкретных x - входных данных для сжатия.

Лучшее качество результатов соответствует экстремуму функции ϕ. Не нарушая общности, будем говорить о минимуме функции.

Функционально оптимальным, с показателем оптимальности ρ, будем считать алгоритм A_0, реализующий функцию $f_0(x)$:

$$\forall x \subset X \quad \rho(f_0(x)) = \min_{\forall A_i} \rho(f_i(x)), \quad (6)$$

где A_i - алгоритм, соответствующий той же спецификации (далее альтернативный), реализующий функцию $f_i(x)$.

Можно говорить о функциональной эффективности алгоритма в сравнении с эталоном – абсолютной функциональной эффективности либо по отношению к другому альтернативному алгоритму – относительной функциональной эффективности.

Абсолютная функциональная эффективность при фиксированном x

$$AFE\big|x = \frac{\rho(f(x)) - \rho(f_0(x))}{\rho(f(x))} \cdot 100\%. \quad (7)$$

определяет, насколько алгоритм, реализующий функцию (1), далек от оптимального.

Оставляя открытыми вопросы существования минимума, функции и алгоритма, удовлетворяющего (6), отметим серьезные практические сложности разработки оптимального алгоритма и доказательства его оптимальности.

Гораздо большую практическую ценность и имеет относительная функциональная эффективность, определяющая, насколько один алгоритм превосходит другой.

Относительная функциональная эффективность двух алгоритмов, реализующих функции $f_1(x)$ и $f_2(x)$ при фиксированном x

$$OFE\big|x = \frac{\rho(f_1(x)) - \rho(f_2(x))}{\max(\rho(f_1(x)), \rho(f_2(x)))} \cdot 100\%. \quad (8)$$

Учитывая гораздо большую востребованность относительного подхода, под функциональной эффективностью, если специально не оговорено, будем иметь в виду относительную функциональную эффективность.

Для сравнения функциональной эффективности двух алгоритмов (аналогично временной эффективности [9]) необходимо определить, насколько в среднем один алгоритм превосходит другой и насколько часто его превосходство.

Аналогично [9] предлагаются два показателя функциональной эффективности алгоритмов:

- степень превосходства одного (i-го) алгоритма над другим (j-м) на ограниченном множестве $\overline{X} \subseteq X$ есть

$$SUP_{ij}\Big|\overline{X} = \frac{1}{N} \sum_{x \subset \overline{X}} \frac{\rho(f_i(x) - \rho(f_j(x))}{\max(\rho(f_i(x), \rho(f_j(x))}, \quad (9)$$

где N - общее количество слагаемых;

- область превосходства

$$REG_{ij}\Big|\overline{X} = \frac{1}{N} \sum_{x \subset \overline{X}} sign(\rho(f_i(x) - \rho(f_j(x))). \quad (10)$$

Следует обосновать возможность суммирования по всем x произвольного множества $\overline{X}$.

Элементами множества $\overline{X}$ могут быть целые, вещественные числа либо конечные структуры из них. При практическом применении алгоритмов указанные числа ограничены некоторым интервалом. Более того, для эффективного моделирования данных программными средствами интервалы значений входных данных следует указывать в спецификации программ.

Количество знаков в представлении модельных вещественных чисел в цифровых ЭВМ ограничено, поэтому и количество таких чисел ограничено некоторым интервалом.

Так как из множества целых и модельных вещественных чисел, как и их комбинации, формируется конечное множество $\overline{X}$, то и само множество $\overline{X}$ конечно.

Применительно к нашему примеру с алгоритмами сжатия функциональная эффективность по показателю оптимальности "объем сжатых данных" определяется степенью превосходства i-го алгоритма над j-м SUP_{ij} и областью превосходства REG_{ij} на множестве $\overline{X}$.

Заметим, что степень сжатия данных не может считаться показателем функциональной эффективности алгоритмов сжатия, так как, с одной стороны, это точечная оценка и не является обобщенной, с другой – не позволяет решать задачи выбора и оценки алгоритма.

Алгоритм может быть применен к различным данным, в том числе и к данным разных типов. Так, алгоритмы кластеризации могут работать в любом метрическом пространстве. Возникает вопрос о функциональной эффективности алгоритмов применительно к данным различных типов.

В этом случае показатели функциональной эффективности SUP_{ij} и REG_{ij} могут определяться в каждом пространстве отдельно:

$$SUP_{ij}\Big|\overline{X}_1 \cdots SUP_{ij}\Big|\overline{X}_K;$$
$$REG_{ij}\Big|\overline{X}_1 \cdots REG_{ij}\Big|\overline{X}_K \quad (11)$$

или для всех типов вместе:

$$SUP_{ij}\Big|\overline{X}_1 \cup \overline{X}_2 \cup \cdots \overline{X}_k$$
$$REG_{ij}\Big|\overline{X}_1 \cup \overline{X}_2 \cup \cdots \overline{X}_k, \quad (12)$$

где $\overline{X}_k$ - множества различных метрических пространств.

В (9)-(12) подразумевается, что управляющие данные являются частью исходных данных, т.е. алгоритм реализует функцию

$$y = f(\tilde{x}), \quad (13)$$

где $\tilde{x} = (x,u) \in X$.

Интерес вызывает функциональная эффективность алгоритмов с оптимальным управлением. При этом оцениваются потенциальные возможности алгоритмов или возможности адаптации к данным посредством управления.

Потенциальная степень превосходства одного (*i*-го) алгоритма над другим (*j*-м) на ограниченном множестве $\overline{X} \subseteq X$ есть

$$SUP_{ij}\Big|\overline{X} = \frac{1}{N}\sum_{x\subset\overline{X}} \frac{l_i(x) - l_j(x)}{\max(l_i(x), l_j(x))}, \quad (14)$$

где $l_i(x) = \min_{u_i}(\rho(f_i(x)))$.

Область потенциального превосходства

$$REG_{ij}\Big|\overline{X} = \frac{1}{N}\sum_{x\subset\overline{X}} sign(l_i(x) - l_j(x)). \quad (15)$$

4. *S*-*R*- показатели функциональной эффективности алгоритмов

Показатели функциональной эффективности SUP_{ij} и REG_{ij} ввиду очень больших вычислительных затрат и значительных людских ресурсов необходимых для их определения практически не применимы.

S-*R*-показатели являются их статистической оценкой. Для их определения сумма заменяется интегралом соответствующей ступенчатой функции, а для вычисления применяется численный метод Монте-Карло [10].

Математическое обоснование этих статистических оценок аналогично временной эффективности алгоритмов [9]. Здесь приведем лишь результаты.

S-оценку степени превосходства *i*-го алгоритма над *j*-м (9) определим как

$$S_{ij}\Big|\overline{X} = \frac{1}{M}\sum_{m=1}^{M} \frac{\rho(f_i(x_m) - \rho(f_j(x_m)}{\max(\rho(f_i(x_m), \rho(f_j(x_m))} * 100\%, \quad (16)$$

где $x_m \subset \overline{X}$ элементы случайным образом сформированной выборки; M – количество экспериментов.

R-оценку области превосходства i-го алгоритма над *j*-м (10) определим как

$$R_{ij}\Big|\overline{X} = \frac{1}{M}\sum_{m=1}^{M} sign(\rho(f_i(x_m) - \rho(f_j(x_m)) * 100\%. \quad (17)$$

Доверительный интервал S-оценки при коэффициенте доверия β:

$$S_{ij} - t_{2,\beta}\sqrt{\frac{1}{6}\sum_{k=1}^{3}(\zeta_k - S_{ij}/100)^2} * 100,$$
$$< S_{ij}\Big|\overline{X} < \quad (18)$$
$$S_{ij} + t_{2,\beta}\sqrt{\frac{1}{6}\sum_{k=1}^{3}(\zeta_k - S_{ij}/100)^2} * 100,$$

где $t_{2,\beta}$- табличное значение распределения Стьюдента;

$$\zeta_k = \frac{1}{3M}\sum_{m=(k-1)M+1}^{kM} \frac{\rho(f_i(x_m) - \rho(f_j(x_m)}{\max(\rho(f_i(x_m), \rho(f_j(x_m))}.$$

Аналогично определяются доверительные интервалы для *R*-показателя.

Таким образом, для определения *S*- и *R*-показателей необходимо:

- выполнить последовательность численных экспериментов. Случайным образом формируется M исходных данных. При этих исходных данных выполняются *i*-й и *j*-й алгоритмы. Определяются результаты их выполнения:
- согласно (16) вычисляется *S*- показатель и (18) его доверительные интервалы;
- согласно (17) вычисляется *R*- по-

казатель и, соответственно, доверительные интервалы.

Выводы

1. Многие эвристические алгоритмы, в основном связанные с направлением программирования, определяемым как искусственный интеллект, представляются как нечетко специфицированные.
2. Функциональная эффективность алгоритмов является объективной эксплуатационной характеристикой нечетко специфицированных алгоритмов.
3. При выборе рациональных алгоритмов для разрабатываемых программных средств следует учитывать их эксплуатационные характеристики. При этом будут эффективными *S*-*R*-показатели, определенные в области, соответствующей их спецификации.
4. *S*-*R*-показатели функциональности алгоритмов позволяют оценить эффективность вновь разработанных алгоритмов по сравнению с существующими аналогами и управлять процессами совершенствования алгоритмов.
5. Предложенные статистические методы определения *S*-*R*-показателей позволяют оценить их достоверность.

1. *Цейтлин Г.Е.* Введение в алгоритмику. - Киев: Сфера, 1998. – 310 с.
2. *Успенский В.А., Семенов А.Л.* Теория алгортмов: основные открытия и приложения. - М.: Наука, 1987. – 288 с.
3. *Агафонов В.Н.* Спецификация программ: понятийные средства и их организация. – Новосибирск: Наука. Сиб. отд-ние, 1990. – 224 с.
4. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео / *Д. Ватолин, А. Ратушняк, М. Смирнов, В. Юкин.* - М.: Диалог-МИФИ, 2002. – 384 с.
5. *Лисков Б., Гатэг Дж.* Использование абстракций и спецификаций при разработке программ. - М.: Мир, 1989. – 424 с.
6. *Гвишиади А.Д., Агаян С.М., Богоутдинов Ш.Р.* Математические методы геоинформатики. I. О новом подходе к кластеризации // Кибернетика и системный анализ. - №2. – С. 104-122.
7. *Ту Дж., Гонсалес Р.* Принципы распознавания образов. - М.: Мир, 1978. – 411с.
8. *Жамбю М.* Иерархический кластер-анализ и соответствия. - М.: Финансы и статистика, 1988. - 342с.
9. *Шинкаренко В.И.* Сравнительный анализ временной эффективности функционально эквивалентных алгоритмов // Пробл. программирования. – 2001. - №3-4 – С. 31-39.
10. *Соболь И.М.* Численные методы Монте-Карло: М. Наука, 1973. – 311 с.
11. *Касперски К.* Техника оптимизации программ. Эффективное использование памяти. СПб. - БХВ-Петербург, 2003. – 464 с.

Получено 26.09.05

Сведения об авторе
Шинкаренко Виктор Иванович
доцент кафедры Компьютерные информационные технологии, канд. техн. наук

Место работы автора
Днепропетровский национальный университет железнодорожного транспорта им. академика В. Лазаряна
Днепропетровск, ул. Акад. Лазаряна, 2,
Тел.8-056-776-19-52 (раб)
Тел. 8-0562-47-39-67 (дом)
E-mail ccp@diit-70.diit.ua,
shink@tk.diit.edu.ua