МАТЕМАТИЧЕСКИЕ ОСНОВЫ ИНФОРМАТИКИ И ПРОГРАММИРОВАНИЯ

DOI 10.17223/20710410/12/3
УДК 510.52+004.051

FPT-АЛГОРИТМЫ И ИХ КЛАССИФИКАЦИЯ НА ОСНОВЕ ЭЛАСТИЧНОСТИ

В. В. Быкова

Институт математики Сибирского федерального университета, г. Красноярск, Россия

E-mail: bykvalen@mail.ru

Приведены основные положения и проблемы параметризированной алгоритмики — нового направления теории сложности вычислений. Предложен новый показатель вычислительной сложности параметризированного алгоритма, с помощью которого можно измерять темп роста функции сложности многих переменных. Этим показателем является частная эластичность функции сложности. Предложена двумерная классификация параметризированных алгоритмов для мультипликативной формы представления функций сложности.

Ключевые слова: *сложность вычислений, параметризированные алгоритмы, анализ алгоритмов, эластичность алгоритмов.*

С позиции классической теории сложности большинство задач, имеющих важное практическое значение, NP-трудные [1]. Классическая теория сложности анализирует и классифицирует задачи и алгоритмы по объему ресурса (преимущественно времени), необходимого для достижения требуемого результата, и потому оперирует функциями вычислительной сложности $t(n)$, зависящими от одной переменной n — длины входа алгоритма. Такая одномерность и ориентация в анализе алгоритмов на худший случай зачастую делает задачи сложнее, чем они есть на самом деле. Для практического решения NP-трудных задач были предложены многие подходы, среди которых приближенные и параметризированные алгоритмы. Цель приближенных алгоритмов — найти за полиномиальное время решение, близкое к оптимальному. Заметный класс приближенных алгоритмов составляют полностью полиномиальные схемы приближений [2]. Параметризированные алгоритмы направлены на поиск точных решений NP-трудных задач, когда параметр k решаемой задачи мал по сравнению с длиной входа алгоритма. Роль этого параметра — учесть информацию о структуре входных данных алгоритма и выделить основной источник неполиномиальной сложности NP-трудной задачи. В параметризированной теории используются двумерные функции сложности алгоритмов (функции, зависящие от n и k), поэтому ее называют двумерной теорией сложности вычислений [3]. Параметризированная теория сложности создает основу для анализа и детальной классификации задач, которые труднорешаемые в классическом понимании, а также для развития новых методов анализа и разработки эффективных алгоритмов решения NP-трудных задач. Идея параметризированного подхода к оценке сложности алгоритмов и задач была предложена в 90-х годах прошлого столетия [4]. За последние два десятилетия эта идея нашла применение в различных областях компьютерных наук, таких, как криптография [5], искусственный интеллект [6], вычислительная биология [7], теория баз данных [8].

1. Параметризированные задачи и алгоритмы

Приведем основные положения теории параметризированной сложности, используя обозначения и понятия, введенные в работах [3, 4]. Параметризированная задача Π состоит в том, что для заданных языка $L(\Pi) \subseteq \Sigma^* \times \mathbb{N}$, где Σ — некоторый конечный алфавит и $\mathbb{N}$ — множество всех неотрицательных целых чисел, и пары $(I, k) \in \Sigma^* \times \mathbb{N}$ требуется определить, является ли (I, k) элементом $L(\Pi)$. Для алгоритма α, решающего данную задачу, (I, k) называют входом, I — его основной частью, $n = |I|$ — длиной входа и число k — параметром задачи. Алгоритм α называют параметризированным алгоритмом, если его вычислительная сложность, определяемая количеством времени исполнения, оценивается с точки зрения длины входа $|I| = n$ и значения параметра k. Таким образом, функция вычислительной сложности параметризированного алгоритма, определяющая время его работы, — функция двух переменных $t(n, k)$.

Параметризированная задача Π считается разрешимой с фиксированным параметром, или FPT-разрешимой (*Fixed-Parameter Tractable*), если она может быть решена некоторым параметризированным алгоритмом за время

$$t(n, k) = \mathrm{O}(n^{\mathrm{O}(1)} \cdot f(k)) \tag{1}$$

для функции f, зависящей только от параметра k. Порядок роста функции $f(k)$ не ограничивается. Так, возможно $f(k) = 2^{\mathrm{o}(k)}$ или $f(k) = 2^{\mathrm{O}(k)}$. Важно, что исключаются функции вида $f(n, k)$, например $f(n, k) = n^k$. Класс всех разрешимых с фиксированным параметром задач обозначается FPT. Соответствующие параметризированные алгоритмы, решающие такие задачи, называются FPT-алгоритмами.

Примером FPT-разрешимой задачи может служить задача о вершинном покрытии графа, когда параметром выступает размер покрытия. В самом деле, вершинное покрытие размера k в графе G с n вершинами можно определить за время $\mathrm{O}(n \cdot 2^k)$. Если в данной задаче в качестве параметра взять древовидную ширину $\mathrm{tw}(G)$ графа G, то метод динамического программирования способен отыскать наибольшее вершинное покрытие за время $\mathrm{O}(n \cdot 2^{\mathrm{tw}(G)})$ [9]. Следовательно, параметризация данной задачи относительно $\mathrm{tw}(G)$ вновь приводит к FPT-разрешимости. Примером задачи, которая не принадлежит FPT, является задача о k-раскраске n-вершинного графа с параметром k.

Очевидно, что в FPT лежат все полиномиально разрешимые задачи. Однако наибольший интерес с точки зрения теории и практики представляют FPT-разрешимые задачи, являющиеся NP-трудными. С теоретической точки зрения все эти задачи могут быть решены за полиномиальное время при каждом фиксированном значении параметра. Реально это удается осуществить в большинстве случаев лишь при малых значениях параметра (все зависит от вида функции $f(k)$). Уже имеются сборники параметризированных задач, включая FPT-разрешимые задачи. Наиболее конкретным на данный момент времени является сборник М. Чезати [10]. Теория параметризированной сложности развивается по нескольким направлениям: определение иерархии классов сложности параметризированных задач, установление условий FPT-разрешимости, выявление взаимосвязи между параметризированной сложностью и классами приближенных алгоритмов (в частности, полностью полиномиальными схемами приближений), развитие параметризированной алгоритмики (методов анализа и разработки параметризированных алгоритмов) [3]. Последнее направление — это преимущественно область прикладных исследований. Здесь имеется ряд открытых вопросов.

2. Некоторые вопросы параметризированной алгоритмики

Анализ сборника параметризированных задач [10] показывает, что одна и та же задача может иметь различные параметризации (различные параметризированные формулировки) относительно разнообразных параметров. Так, в задаче о вершинном покрытии параметрами могут быть размер покрытия, число вершин, число ребер, древовидная ширина, наибольшая степень вершин графа и др. Закономерны вопросы:

1) Что может служить в качестве параметра для NP-трудной задачи?
2) Как следует выбирать параметр, чтобы задача стала FPT-разрешимой?
3) Существуют ли специальные методы разработки FPT-алгоритмов?
4) Как сравнивать между собой различные FPT-алгоритмы, предназначенные для решения одной и той же NP-трудной задачи?

На некоторые из этих вопросов пока нет исчерпывающих ответов. Например, на первые два вопроса параметризированная теория сложности отвечает следующим образом.

Бывают параметры внутренние и внешние. Параметр считается явным внутренним, если он непосредственно появляется в основной части экземпляра (I, k) параметризированной задачи П и его значение ограничено полиномом от $|I| = n$. Для задачи о вершинном покрытии подобными параметрами являются размер покрытия, число вершин или число ребер графа. Имеются стандартные параметры. Они наиболее популярны. Так, в задачах минимизации в роли стандартного параметра всегда выступает величина, которая минимизируется. Для задачи о вершинном покрытии — это размер покрытия. Если $k = n$, то k становится тривиальным параметром, и параметризированный алгоритм вырождается в обычный алгоритм. Параметр является неявным внутренним, если его значение ограничено полиномом от $|I| = n$ и информация о нем неявно скрыта в I. Типичным примером неявного внутреннего параметра является древовидная ширина графа (значение этой величины никогда не превосходит числа вершин графа). В целом, все внутренние параметры характеризуют некоторым образом структуру входных данных алгоритма. Наконец, существуют внешние параметры, которые никак не связаны с $|I|$ или связаны с $|I|$ неполиномиальным образом. Информация о таких параметрах задается дополнительно к I, так как считается, что ее нет в I.

С теоретической точки зрения в качестве параметра может выступать любая часть входа алгоритма. На практике для получения хорошей параметризации (например, относящейся к FPT) рекомендуется выбирать ту часть входа, которая обычно принимает малые значения по сравнению с $|I| = n$. В конечном счете, все зависит от приложения. Для различных приложений могут быть приемлемы различные параметризации одной и той же NP-трудной задачи. Относительно ряда NP-трудных задач доказано, что некоторые их параметризации не могут быть реализованы FPT-алгоритмами. В частности, такой задачей является задача о k-раскраске n-вершинного графа с параметром k. Для некоторых NP-трудных задач найдены FPT-алгоритмы лишь применительно к отдельным параметрам. Например, для многих оптимизационных задач на графах древовидная ширина оказалась таким параметром, который, как правило, приводит к FPT-алгоритмам [9 – 11].

Что касается третьего из перечисленных выше вопросов, то в настоящее время уже предложены и апробированы несколько специальных методов конструирования FPT-алгоритмов. Прежде всего, это метод динамического программирования на основе дерева декомпозиции для графов с ограниченной древовидной шириной [11]. Другим

известным методом построения FPT-алгоритмов является параметрическая редукция (*kernelization*) [4]. Параметрическая редукция — это полиномиальное по времени преобразование каждого экземпляра (I, k) параметризированной задачи Π в экземпляр (I', k') параметризированной задачи Π' так, что $k' < k$. После выполнения редукции к экземпляру (I', k'), называемому ядром, применяется полный перебор. Например, для задачи о вершинном покрытии со стандартным параметром k (размером покрытия) редукция сводится к многократному применению двух правил: удаление изолированной вершины без изменения значения параметра k; удаление вершины, степень которой больше k, и уменьшение параметра k на единицу. Следует отметить, что метод параметрической редукции в сочетании с полным перебором не всегда приводит к FPT-алгоритму. Между тем доказано, что если параметризированная задача принадлежит к FPT, то данный метод неизменно дает FPT-алгоритм [4].

Четвертый вопрос теоретически открыт: в многочисленных работах по параметризированной теории сложности для анализа и сравнения параметризированных алгоритмов применяются методы классической (одномерной) теории сложности [12 – 14]. Между тем классическая одномерность ограничивает глубину анализа параметризированных алгоритмов, для которых вычислительная сложность описывается функцией от двух переменных $t(n, k)$. В настоящей работе предлагается мера вычислительной сложности алгоритма, с помощью которой можно исследовать темп роста функций многих переменных, анализировать степень влияния структуры входных данных параметризированного алгоритма на его вычислительную сложность, сравнивать между собой FPT-алгоритмы решения NP-трудных задач. Этой мерой является частная эластичность функции вычислительной сложности (далее просто функции сложности) алгоритма. Представленные в статье результаты являются обобщением и расширением результатов автора, опубликованных в [15 – 17] применительно к одномерной теории сложности вычислений.

3. Соглашения относительно функций сложности

Формальный подход к анализу параметризированных алгоритмов требует уточнения свойств их функций сложности $t(n, k)$. Относительно функций $t(n, k)$ мы сделаем несколько естественных допущений, которые выполнимы для большинства реальных алгоритмов:

- полагаем, что $t(n, k)$ — монотонно неубывающие функции по обоим аргументам, областью значений функций выступает множество неотрицательных действительных чисел, а областью определения — множество $\mathbb{N} \times \mathbb{N}$;
- допускаем возможность отступления от дискретности изменения n и k (с формальной заменой n на x и k на y), т. е. считаем, что аргументы функции $t(n, k)$ непрерывны, а необходимые значения вычисляются в целочисленных точках $x = n$ и $y = k$. Руководствуясь данным допущением, ниже вместо $t(n, k)$ будем писать $z(x, y)$;
- предполагаем, что рассматриваемое множество функций ограничивается семейством $\mathfrak{L}$ «по-существу положительных», логарифмически-экспоненциальных функций. Функция $z(x, y)$ считается «по-существу положительной», если существуют такие значения x_0, y_0, что $z(x, y) > 0$ для всех $x > x_0$, $y > y_0$. Семейство $\mathfrak{L}$ определено рекурсивно, и всякая функция $z(x, y) \in \mathfrak{L}$ представима в виде $z(x, y) = e^{w(x,y)}$, где $w(x, y) \in \mathfrak{L}$ [18]. Известно, что каждая такая функция непрерывна и дифференцируема в той области, где она определена.

Данные предположения являются гарантией существования эластичности для функций сложности параметризированных алгоритмов.

4. Эластичность функции как мера вычислительной сложности алгоритма

Под эластичностью $E_x(z)$ функции $z = z(x)$ принято понимать предел отношения относительного приращения этой функции к относительному приращению аргумента:

$$E_x(z) = \lim_{\Delta x \to 0} \left(\frac{\Delta z}{z} : \frac{\Delta x}{x} \right) = \frac{dz}{z} : \frac{dx}{x} = \frac{dz}{dx} \cdot \frac{x}{z} = z' \frac{x}{z} = x(\ln z)'. \tag{2}$$

Отметим несколько важных особенностей эластичности $\mathfrak{L}$-функции. Для всякой $\mathfrak{L}$-функции $z = z(x)$ всегда существует эластичность и $E_x(z) \geqslant 0$. В [15] показано, что при $x \to \infty$ эластичность $\mathfrak{L}$-функции — тождественно постоянная, или бесконечно малая, или бесконечно большая логарифмически-экспоненциальная функция, для которой

$$E_x(z) = \mathrm{o}\left(\frac{z}{\ln x}\right), \quad E_x(cz^m) = \mathrm{O}(E_x(z)), \quad c > 0, m > 0.$$

Последняя оценка означает, что при $x \to \infty$ поведение эластичности инвариантно относительно полиномиального преобразования, заключающегося в умножении этой функции на константу $c > 0$ и возведении в положительную и отделенную от нуля степень $m > 0$. Хорошо известно [18], что две произвольные $\mathfrak{L}$-функции $z_1(x)$, $z_2(x)$ всегда сопоставимы между собой по порядку роста: если $z_1(x), z_2(x) \in \mathfrak{L}$, то при $x \to \infty$ верно одно из трех отношений:

$$z_1(x) \prec z_2(x), \quad z_2(x) \prec z_1(x), \quad z_1(x) = \mathrm{O}(z_2(x)).$$

Кроме того, доказано, что иерархия эластичностей порождает идентичную иерархию $\mathfrak{L}$-функций [17], т. е. если $E_x(z_1) \prec E_x(z_2)$, то $z_1(x) \prec z_2(x)$. Заметим, что здесь отношение $\prec$ интерпретируется так: $z_1(x) \prec z_2(x)$ тогда и только тогда, когда $z_1(x) = \mathrm{o}(z_2(x))$. О-большое обозначает асимптотическую пропорциональность функций: $z_1(x) = \mathrm{O}(z_2(x))$ тогда и только тогда, когда $z_1(x) \sim cz_2(x)$, $c > 0$. Согласно (2), при достаточно малых Δx справедливо приближение

$$\frac{\Delta z}{z} \approx E_x(z) \frac{\Delta x}{x},$$

которое указывает, что $E_x(z)$ — коэффициент пропорциональности между темпами роста переменных z и x. Поскольку $E_x(z) = E_{ax}(bz)$ для любых констант $a > 0$, $b > 0$, то эластичность — безразмерная величина.

Все перечисленные выше особенности эластичности позволяют использовать ее в качестве меры измерения вычислительной сложности алгоритма, поскольку она удовлетворяет ряду необходимых требований, таких, как сопоставимость, интерпретируемость и вычислимость. Сопоставимость обеспечивают безразмерность и инвариантность относительно модели вычислений: ведь переход от одной модели к другой меняет вычислительную сложность алгоритма лишь полиномиальным образом. Эластичность имеет отчетливую интерпретацию: если $z = z(x)$ — функция сложности алгоритма, то это означает, что при повышении значения x (длины входа алгоритма) на один процент значение z (время выполнения алгоритма) увеличивается приблизительно на $E_x(z)$ процентов. Вычислимость также имеет место: эластичность существует для всякой $\mathfrak{L}$-функции и всегда может быть определена непосредственно из (2) и свойств эластичности. Эластичности присущи свойства, сходные со свойствами операций логарифмирования и дифференцирования [16], поэтому она легко определяется для логарифмически-экспоненциальных функций. Простота вычисления — это основное достоинство эластичности по сравнению с другими мерами сложности вычислений.

Важно отметить, что эластичность — локальная характеристика функции: ее значения в общем случае изменяются при переходе от одного значения аргумента к другому. Между тем при больших значениях аргумента в поведении эластичности $\mathfrak{L}$-функций наблюдаются определенные закономерности: разным (по скорости роста) классам $\mathfrak{L}$-функций свойственно принципиально различное поведение эластичности, и наоборот, по асимптотике поведения эластичности можно однозначным образом определить класс, к которому относится $\mathfrak{L}$-функция. Это утверждает следующая теорема.

Теорема 1 (о классификации $\mathfrak{L}$-функций) [15]. Разбиение семейства монотонно неубывающих «по-существу положительных» $\mathfrak{L}$-функций на классы SUBPOLY, POLY, SUBEXP, EXP, HYPEREXP в соответствии с порядком их роста эквивалентно надлежащему разбиению по асимптотике эластичности этих функций на бесконечности:

$$\text{SUBPOLY} = \{z(x) : z(x) \prec e^{\mathrm{O}(\ln x)}\} \equiv \{z(x) : E_x(z) = \mathrm{o}(1)\}; \tag{3}$$

$$\text{POLY} = \{z(x) : z(x) = \mathrm{O}\left(e^{\mathrm{O}(\ln x)}\right)\} \equiv \{z(x) : E_x(z) = \mathrm{O}(1)\}; \tag{4}$$

$$\text{SUBEXP} = \{z(x) : e^{\mathrm{O}(\ln x)} \prec z(x) \prec e^{\mathrm{O}(x)}\} \equiv \{z(x) : 1 \prec E_x(z) \prec x\}; \tag{5}$$

$$\text{EXP} = \{z(x) : z(x) = \mathrm{O}\left(e^{\mathrm{O}(x)}\right)\} \equiv \{z(x) : E_x(z) = \mathrm{O}(x)\}; \tag{6}$$

$$\text{HYPEREXP} = \{z(x) : e^{\mathrm{O}(x)} \prec z(x)\} \equiv \{z(x) : x \prec E_x(z)\}. \tag{7}$$

Данная теорема порождает пять сложностных классов алгоритмов (субполиномиальных, полиномиальных, субэкспоненциальных, экспоненциальных и гиперэкспоненциальных соответственно), которые полностью отвечают современной классификации алгоритмов. Класс быстрых алгоритмов составляют алгоритмы с функциями сложности $z(x) \in \text{SUBPOLY}$. Таким алгоритмам присуща тождественно нулевая или бесконечно малая эластичность. Класс полиномиальных алгоритмов — множество алгоритмов с $z(x) \in \text{POLY}$ и асимптотически постоянной эластичностью $E_x(z)$. Класс субэкспоненциальных алгоритмов — алгоритмы, для которых $z(x) \in \text{SUBEXP}$. Эластичность $E_x(z)$ субэкспоненциального алгоритма — бесконечно большая величина, такая, что $1 \prec E_x(z) \prec x$. Класс экспоненциальных алгоритмов — это алгоритмы, для которых $z(x) \in \text{EXP}$. Для них эластичность $E_x(z) = O(x)$ — бесконечно большая величина, асимптотически пропорциональная линейной функции. Класс гиперэкспоненциальных алгоритмов — это алгоритмы, для которых $z(x) \in \text{HYPEREXP}$ и $x \prec E_x(z)$. Эквивалентности (3) – (7) обеспечивают «прозрачность» сложностных классов алгоритмов, ведь эластичности согласно (3) – (7) представляются более простыми по виду функциями, чем соответствующие им $\mathfrak{L}$-функции. Понятие эластичности легко распространяется на функции многих переменных. Это дает возможность использовать данную дифференциальную характеристику функции в анализе параметризированных алгоритмов.

Под частной эластичностью $E_x(z)$ функции $z = z(x, y)$ по аргументу x понимается эластичность переменной z, которая рассматривается как функция только от x и при постоянных значениях y. Частная эластичность $E_x(z)$ связана с частной производной функции $z = z(x, y)$ по x соотношением

$$E_x(z) = z'_x \frac{x}{z} = x(\ln z)'_x. \tag{8}$$

Аналогично определяется частная эластичность $E_y(z)$ функции $z = z(x, y)$ по аргументу y:

$$E_y(z) = z'_y \frac{y}{z} = y(\ln z)'_y. \tag{9}$$

Поскольку всякая $\mathfrak{L}$-функция $z = z(x, y)$ непрерывна и дифференцируема в той области, где она определена, то для нее в этой области всегда существуют частные эластичности. Выражения (8), (9) аналогичны формуле (2). Поэтому частные эластичности $E_x(z), E_y(z)$ обладают всеми основными свойствами эластичности функции одной переменной [16]. Укажем наиболее важные из этих свойств и запишем их применительно к $E_x(z)$:

1) Если $z = z(x, y)$ не зависит от x, то $E_x(z) = 0$.
2) Безразмерность: $E_x(z) = E_{ax}(bz)$ для любых констант $a > 0, b > 0$.
3) Частная эластичность произведения (отношения) функций $z_1 = z_1(x, y)$ и $z_2 = z_2(x, y)$ равна сумме (разности) их частных эластичностей:

$$E_x(z_1 \cdot z_2) = E_x(z_1) + E_x(z_2), \quad E_x(z_1/z_2) = E_x(z_1) - E_x(z_2).$$

Частным эластичностям свойственна отчетливая интерпретация. Пусть $z = z(n, k)$ является функцией сложности параметризированного алгоритма, где n — длина входа, а k — параметр этого алгоритма. После формальной замены n на x и k на y имеем $z = z(x, y) \in \mathfrak{L}$. Тогда $E_x(z)$ — коэффициент пропорциональности между темпом роста времени работы алгоритма и темпом роста его длины входа x. Аналогично $E_y(z)$ — коэффициент пропорциональности между темпом роста времени выполнения алгоритма и темпом роста параметра y.

5. Двумерная классификация FPT-алгоритмов по сложности

В общем случае частные эластичности $E_x(z), E_y(z)$ являются функциями, зависящими от двух аргументов x и y. Однако ситуация значительно упрощается, если учесть тот факт, что для большинства параметризированных алгоритмов время работы алгоритма описывается $\mathfrak{L}$-функцией вида

$$z(x, y) = q(x) \cdot f(y), \tag{10}$$

где $q(x) \in \mathfrak{L}$ — количественная компонента, а $f(y) \in \mathfrak{L}$ — параметрическая компонента функции $z(x, y)$. Мультипликативная форма представления (10) характерна для алгоритмов с параметрически зависимым циклом, выполняющим полный перебор всех возможных вариантов при различных значениях параметра, тогда как в теле этого цикла выполняется проверка текущего варианта и время проверки зависит только от длины входа алгоритма. Именно такие алгоритмы порождает метод параметрической редукции. Заметим, что формула (1), определяющая время работы FPT-алгоритма, также имеет мультипликативную форму записи:

$$z(x, y) = \mathrm{O}(x^{\mathrm{O}(1)} \cdot f(y)). \tag{11}$$

Пусть $z(x, y) = q(x) \cdot f(y) \in \mathfrak{L}$. Тогда, исходя из свойств эластичности 1 и 3, частные эластичности $E_x(z), E_y(z)$ вырождаются в обычные эластичности функции одного аргумента:

$$E_x(z) = E_x(q(x) \cdot f(y)) = E_x(q(x)) + E_x(f(y)) = E_x(q(x)); \tag{12}$$
$$E_y(z) = E_y(q(x) \cdot f(y)) = E_y(q(x)) + E_y(f(y)) = E_y(f(y)). \tag{13}$$

Теперь $E_x(z)$ зависит лишь от x, а $E_y(z)$ — только от y. Поскольку $q(x), f(y) \in \mathfrak{L}$, то каждая из этих функций принадлежит только одному сложностному классу из $\mathfrak{K} = \{\text{SUBPOLY, POLY, SUBEXP, EXP, HYPEREXP}\}$. Обозначим через $\mathfrak{F}_x$ класс

сложности функции $z(x,y)$ по аргументу x, а через $\mathfrak{F}_y$ — класс сложности по аргументу y. Тогда всякому параметризированному алгоритму с функцией сложности $z(x,y) = q(x)\cdot f(y) \in \mathfrak{L}$ соответствует пара

$$(\mathfrak{F}_x, \mathfrak{F}_y) \in \mathfrak{K} \times \mathfrak{K},$$

характеризующая сложность данного алгоритма как по длине входа x, так и по значению параметра y. Таким образом, мы приходим к двумерной классификации параметризированных алгоритмов по сложности. При двумерном подходе более отчетливую и общую формулировку получает определение FPT-алгоритма и условие (11): параметризированный алгоритм называется FPT-алгоритмом, если время его исполнения задается функцией $z(x,y) = q(x)\cdot f(y) \in \mathfrak{L}$, для которой

$$(\mathfrak{F}_x, \mathfrak{F}_y) \in \{\text{SUBPOLY, POLY}\} \times \mathfrak{K}.$$

Когда параметризированная задача не только FPT-разрешима, но и полиномиально разрешима, то для нее существуют FPT-алгоритмы, сложность которых соответствует парам

$$(\mathfrak{F}_x, \mathfrak{F}_y) \in \{\text{SUBPOLY, POLY}\} \times \{\text{SUBPOLY, POLY}\}. \tag{14}$$

Подобные параметризированные алгоритмы естественно назвать полиномиальными FPT-алгоритмами. Алгоритмы, сложность которых отвечает парам

$$(\mathfrak{F}_x, \mathfrak{F}_y) \in \{\text{SUBPOLY, POLY}\} \times \{\text{SUBEXP}\}, \tag{15}$$

$$(\mathfrak{F}_x, \mathfrak{F}_y) \in \{\text{SUBPOLY, POLY}\} \times \{\text{EXP}\}, \tag{16}$$

$$(\mathfrak{F}_x, \mathfrak{F}_y) \in \{\text{SUBPOLY, POLY}\} \times \{\text{HYPEREXP}\}, \tag{17}$$

целесообразно определить как субэкспоненциальные, экспоненциальные и гиперэкспоненциальные FPT-алгоритмы соответственно. Такие FPT-алгоритмы присущи NP-трудным задачам.

Введенная двумерная классификация FPT-алгоритмов не противоречит понятиям, используемым в настоящее время в параметризированной теории сложности применительно к FPT-алгоритмам, а лишь формально их уточняет и расширяет. При желании в декартовом произведении $\mathfrak{K}\times\mathfrak{K}$ всегда можно выделить более мелкие подмножества пар $(\mathfrak{F}_x, \mathfrak{F}_y)$ и, как следствие, определить более детальную классификацию параметризированных алгоритмов, чем (14) – (17). При тривиальной параметризации $y = x$ соотношения (12), (13) принимают единый вид

$$E_x(z) = E_x(q(x)\cdot f(x)) = E_x(q) + E_x(f).$$

В этом частном случае приходим к одномерной классификации алгоритмов. Аналогичный случай возникает, когда $f(y)$ — тождественная константа.

Заключение

Параметризированная теория сложности дала толчок к разработке новых подходов конструирования алгоритмов (алгоритмов решения задач, разрешимых с фиксированным параметром). Одновременно с этим, данная теория породила ряд проблем. Одна из них — отсутствие простых и удобных инструментов анализа параметризированных алгоритмов, когда их функции сложности зависят от многих переменных. В настоящей работе предложен показатель, с помощью которого можно измерять темп роста

функций многих переменных, сравнивать между собой FPT-алгоритмы, анализировать степень влияния параметра на вычислительную сложность параметризированного алгоритма. Этим показателем является частная эластичность. Для мультипликативной формы представления функций сложности в работе предложена двумерная классификация параметризированных алгоритмов, точно определен класс FPT-алгоритмов, выделены подклассы полиномиальных, субэкспоненциальных, экспоненциальных и гиперэкспоненциальных FPT-алгоритмов. Допустимо дальнейшее развитие предложенного подхода в части увеличения числа параметров и анализа других форм представления функций сложности.

ЛИТЕРАТУРА

1. *Гэри М., Джонсон Д.* Вычислительные машины и труднорешаемые задачи. М.: Мир, 1982.
2. *Ausiello G., Crescenzi P., Gambosi G., et al.* Complexity and approximation, combinatorial optimization problems and their approximability properties. New York: Springer Verlag, 1999.
3. *Flum J. and Grohe M.* Parameterized complexity theory. Berlin; Heidelberg: Springer Verlag, 2006.
4. *Downey R. and Fellows M.* Parameterized complexity. New York: Springer Verlag, 1999.
5. *Fellows M. and Koblitz N.* Fixed-parameter complexity and cryptography // Technical Report DCS-207-IR, Department of Comput. Science, University of Victoria, 1992.
6. *Gottlob G., Scarcello F., and Sideri M.* Fixed-parameter complexity in AI and Nonmonotonic reasoning // Artific. Intellig. 2002. V. 138. P. 55–86.
7. *Bodlaender H., Downey R., Fellows M., et al.* Parameterized complexity analysis in computational biology // Comput. Appl. Bioscien. 1995. V. 11. P. 49–57.
8. *Papadimitriou C. and Yannakakis M.* On the complexity of database queries // J. Comput. System Scien. 1999. V. 58. P. 407–427.
9. *Niedermeier R.* Invitation to fixed-parameter algorithms: Oxford Lecture series in mathematics and its applications. Oxford: University Press, 2006.
10. *Cesati M.* Compendium of parameterized problems. `http://bravo.ce.uniroma2.it/home/cesati/research/compendium` — 2006.
11. *Gottlob G., Pichler R., and Wei F.* Abduction with bounded treewidth: From theoretical tractability to practically efficient computation // Proc. of the Twenty-Third AAAI Conf. on Artificial Intelligence. Menlo Park: AAAI Press, 2008. P. 1541–1546.
12. *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы: построение и анализ. М.: МЦНМО, 1999.
13. *Грин Д., Кнут Д.* Математические методы анализа алгоритмов. М.: Мир, 1987.
14. *Быкова В. В.* Математические методы анализа рекурсивных алгоритмов // Журн. СФУ. Математика и физика. 2008. № 1(3). С. 236–246.
15. *Быкова В. В.* Метод распознавания классов алгоритмов на основе асимптотики эластичности функций сложности // Журн. СФУ. Математика и физика. 2009. № 2(1). С. 46–61.
16. *Быкова В. В.* Эластичность алгоритмов // Прикладная дискретная математика. 2010. № 2(8). С. 87–95.
17. *Bykova V. V.* Complexity and elasticity of the computation // Proc. of the 3-rd IASTED International Multi-Conference on Automation, Control, and Information Technology (ACIT-CDA 2010). Anaheim-Calgary-Zurich: ACTA Press, 2010. P. 334–340.
18. *Грэхем Р., Кнут Д., Поташник О.* Конкретная математика. М.: Мир; Бином. Лаборатория знаний, 2006.