УДК 519.178

МЕТОДЫ РАЗРАБОТКИ FPT-АЛГОРИТМОВ НА ГРАФАХ ОГРАНИЧЕННОЙ ДРЕВОВИДНОЙ ШИРИНЫ

В. В. Быкова

Параметризированный подход к оценке сложности вычислений — естественный способ справиться с трудноразрешимостью задач, которые охарактеризованы как NP-трудные в соответствии с классической дихотомией **P** против **NP** [1]. Основная идея параметризированного подхода состоит в том, чтобы с помощью некоторого числового параметра учесть структуру исходных данных и выделить основной источник трудноразрешимости задачи. Параметризированный подход позволяет исследовать различные параметризации одной и той же задачи, каждая из которых может приводить или не приводить к FPT-алгоритмам. Такие алгоритмы представляют интерес для алгоритмической практики, так как при ограниченном значении параметра время их выполнения полиномиально зависит от длины входа алгоритма [2].

В настоящее время уже известны некоторые специальные методы проектирования FPT-алгоритмов для задач, решение которых ищется в конечной области [3, 4]. При этом поиск решения подразумевает нахождение одного из допустимых решений (в задачах разрешения и удовлетворения ограничений) или поиск оптимального допустимого решения (в задачах комбинаторной оптимизации). Такие задачи в литературе часто называют задачами выбора. Большинство этих задач формулируются непосредственно как задачи на графах. Другие допускают графовое представление.

В работе рассматриваются три наиболее цитируемых метода разработки FPT-алгоритмов для задач выбора: параметрическая редукция (*kernelization*), поиск по дереву и метод динамического программирования на основе дерева декомпозиции. Суть метода параметрической редукции состоит в полиномиальном по времени преобразовании каждого экземпляра (I_1, k_1) параметризированной задачи П в экземпляр (I_2, k_2) этой задачи так, что $k_2 < k_1$. Результатом выполнения редукции является экземпляр (I_2, k_2) задачи П, называемый ядром. К ядру применяется полный перебор или какой-либо другой точный метод решения. Известно [1], что параметризированная задача является FPT-разрешимой относительно некоторого параметра тогда и только тогда, когда она может быть редуцирована по этому параметру на основе конечного набора правил редукции. 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. Преимущество метода параметрической редукции состоит в том, что он легко программируется, а трудность — нужно располагать набором правил редукции, которые специфичны для каждой отдельной задачи.

Другой известный метод построения FPT-алгоритмов — это поиск по дереву. Суть метода [4]: выделение конечного пространства поиска, размер которого зависит только от значения параметра, и представление этого пространства в виде дерева. При построении дерева поиска надо располагать правилом выделения узлов-потомков. Такие правила, конечно же, характерны для каждой решаемой задачи выбора. Например, для задачи о вершинном покрытии графа $G = (V, E)$ со стандартным параметром k создается бинарное корневое дерево поиска высоты k. Корню этого дерева приписываются в качестве метки множество $Z = \varnothing$ и граф $G' = G$. Затем в графе G' выбирается некоторое ребро $e = \{u, v\}$. Очевидно, что всякое вершинное покрытие для G содержит

либо вершину u, либо вершину v. Исходя из этого, корневой узел порождает два узла-потомка, первый из них получает метку $Z = Z \cup \{u\}$ и $G' = G' - u$, а второй — метку $Z = Z \cup \{v\}$ и $G' = G' - v$. Далее этот процесс повторяется для каждого вновь созданного узла. Таким образом, формируемые в метках узлов множества Z представляют собой возможные вершинные покрытия, а графы — то, что остается от G' для дальнейшего рассмотрения. На каждом уровне дерева поиска размер возможных вершинных покрытий увеличивается ровно на единицу. Любому узлу, который маркирован безреберным графом G', приписывается множество Z, определяющее вершинное покрытие графа G'. Следовательно, если на высоте не более чем k возникает узел с безреберным графом G', то искомое вершинное покрытие — вершинное покрытие размером не более k — найдено, в противном случае такого покрытия для графа G не существует. Алгоритм поиска по дереву затрачивает на свою работу $O(n2^k)$ времени, где $n = |V|$, и потому является FPT-алгоритмом относительно параметра k.

Следует отметить, что методы параметрической редукции и поиска по дереву, вообще говоря, реализуют традиционные стратегии разработки алгоритмов: первый из них воплощает принцип «уменьшай и властвуй», а второй — «разделяй и властвуй». Особенность лишь в том, что здесь они нацелены на получение FPT-алгоритмов. Основной недостаток этих методов — сильная привязка к специфике решаемых задач.

Наиболее универсальный метод разработки FPT-алгоритмов для задач выбора — метод динамического программирования на основе дерева декомпозиции, автором которого считается Г. Бодлаендер [3]. Данный метод — сочетание декомпозиционного подхода к решению задачи выбора с декомпозиционным представлением входного графа, когда выделение подзадач и построение их решений осуществляется, исходя из этого представления. Дерево декомпозиции графа — это специальная конструкция, которая описывает структуру графа «с точностью до клик и сепараторов» и определяет его древовидную ширину [5]. Метод динамического программирования на основе дерева декомпозиции приводит к FPT-алгоритму, если проверяемое свойство графа выражено в виде конечной формулы монадической логики второго порядка и входной граф имеет ограниченную древовидную ширину [6]. В работе подробно исследован этот метод и установлено, что, несмотря на теоретическую привлекательность, практическое его применение ограничивается двумя существующими проблемами, первая из них связана с высокими потребностями в памяти получаемых FPT-алгоритмов, а вторая — с вычислением древовидной ширины и построением дерева декомпозиции. В данной работе предложено проблему памяти решать с помощью бинарного сепараторного дерева декомпозиции (B&S-дерева), которое обеспечивает компромисс между размером и числом таблиц динамического программирования.

Корневое дерево декомпозиции (M, T), $M = \{X_i : i \in I\}$, $T = (I, W)$, графа G называется B&S-деревом, если в нем каждый узел имеет не более двух прямых потомков и относится к одному из четырех типов узлов:

1) узел-лист — узел, у которого нет потомков;
2) узел-сепаратор — узел s с одним прямым потомком j и узлом i в роли родителя. Всегда X_s — сепаратор графа G и $X_s = X_i \cap X_j \neq \varnothing$, $X_s \subseteq X_i$, $X_s \subseteq X_j$;
3) узел-расширение — узел i с одним прямым потомком s. В данном случае $X_s \subseteq X_i$;
4) узел-соединение — узел i с двумя прямыми потомками l и j. Здесь $X_l \cup X_j \subseteq X_i$.

В работе доказано, что всякое фундаментальное дерево декомпозиции n-вершинного графа, имеющее ширину k и $O(n)$ узлов, может быть преобразовано за время $O(n)$

в B&S-дерево, которое обладает той же шириной и $O(n)$ узлами. Теоретически обоснованы следующие достоинства B&S-дерева: переход от бинарного корневого дерева декомпозиции к B&S-дереву увеличивает число таблиц не более чем в два раза; B&S-дерево позволяет удерживать размер всякой таблицы динамического программирования в пределах оценки $O(kq^k)$, где q — число состояний, в которых может находиться вершина графа по отношению к возможному решению. Кроме того, для вычисления таблиц динамического программирования по B&S-дереву возможно привлечение аппарата реляционной алгебры и свойств ациклических баз данных. Это способствует алгоритмической конкретизации метода и сокращению числа строк в промежуточных таблицах (за счет использования монотонного плана соединения и программы полусоединений таблиц). Приведена демонстрация применения метода динамического программирования по B&S-дереву при решении оптимизационных задач (на примере задачи о вершинном покрытии) и задач удовлетворения ограничений (на примере задачи SAT).

Подробное изложение представленных результатов можно найти в [7].

ЛИТЕРАТУРА

1. *Downey R. and Fellows M.* Parameterized complexity. New York: Springer Verlag, 1999.
2. *Быкова В. В.* FPT-алгоритмы и их классификация на основе эластичности // Прикладная дискретная математика. 2011. № 2(12). С. 40–48.
3. *Bodlaender H. L.* Treewidth: Algorithmic techniques and results // LNCS. 1997. V. 1295. P. 19–36.
4. *Niedermeier R. and Rossmanith P.* A general method to speed up fixed-parameter-tractable algorithms // Inform. Proc. Lett. 2000. V. 73. P. 125–129.
5. *Быкова В. В.* Вычислительные аспекты древовидной ширины графа // Прикладная дискретная математика. 2011. № 3(13). С. 65–79.
6. *Courcelle B.* The monadic second-order logic of graphs. III. Tree decompositions, minors and complexity issues // RAIRO Inform. Theor. Appl. 1992. No. 26(3). P. 257–286.
7. *Быкова В. В.* FPT-алгоритмы на графах ограниченной древовидной ширины // Прикладная дискретная математика. 2012. № 2(16). С. 65–78.

УДК 004.65

ЭЛЕМЕНТЫ РАСПРЕДЕЛЁННОЙ СУБД НА БАЗЕ СЕРВЕРА MariaDB[1]

И. Н. Глотов, С. В. Овсянников, В. Н. Тренькаев

В связи с интенсивным развитием и появлением новых интернет-технологий, таких, например, как «облачные» вычисления, остаётся актуальной задача распределённой обработки больших объёмов данных в режиме реального времени. Так, имеется необходимость в распределённой СУБД, т. е. системе управления, распределённой в компьютерной сети базы данных (БД) [1]. При этом распределённая БД должна обеспечивать прозрачный доступ к данным, т. е. должна выглядеть с точки зрения пользователей и прикладных программ как централизованная БД.

Наиболее востребованными свойствами распределённых СУБД, ориентированных на использование в «облачных» сервисах, для которых характерна работа с большими

[1]Работа выполнена в рамках реализации соглашения о сотрудничестве между кафедрой защиты информации и криптографии ТГУ и ООО «Хайвекс Технолоджи» по проекту JelasticDB.