Глава 4

Вероятностные алгоритмы и их анализ

4.1 Вероятностная проверка тождеств

Один из первых примеров вероятностных алгоритмов, более эффективных, чем детерминированные, был предложен Фрейвалдом для задачи проверки матричного равенства $AB = C$ (см. [Fre77]).

Обычный детерминированный алгоритм заключается в перемножении матриц A и B и сравнении результата с C. Сложность такого алгоритма есть $O(n^3)$ при использовании стандартного алгоритма умножения матриц размера $n \times n$ и $O(n^{2.376})$ при использовании лучшего из известных быстрых алгоритмов матричного умножения [CW90].

Вероятностный алгоритм Фрейвалда для этой задачи имеет сложность $O(n^2)$ и заключается в умножении левой и правой частей на случайный булев вектор $\mathbf{x} = (x_1, \ldots, x_n)$ с последующим сравнением полученных векторов. Алгоритм выдает ответ, что $AB = C$, если $ABx = Cx$, и является алгоритмом с од-

носторонней ошибкой, т. е. ошибается только для случаев неравенства — легко видеть, что если алгоритм сообщает, что тождество не выполнено, то всегда $AB \neq C$.

При этом ABx вычисляется как $A(Bx)$, что и обеспечивает оценку сложности алгоритма $O(n^2)$.

Следующая теорема обеспечивает корректность алгоритма.

Теорема 13. *Пусть A, B, и C — $n \times n$ матрицы, элементы которых принадлежат некоторому полю* **F**, *причем $AB \neq C$. Тогда для вектора x, выбранного случайно и равномерно из $\{0,1\}^n$,*

$$P\{ABx = Cx\} \leq 1/2.$$

Доказательство. Пусть $D = AB - C$. Мы знаем, что D — не полностью нулевая матрица и хотим оценить вероятность того, что $D\mathbf{x} = 0$. Без ограничения общности можно считать, что ненулевые элементы имеются в первой строке и они располагаются перед нулевыми. Пусть **d** — вектор, равный первой строке матрицы D, и предположим, что первые k элементов в **d** — ненулевые. Имеем

$$\mathsf{P}\{D\mathbf{x} = 0\} \leq \mathsf{P}\{\mathbf{d}^T\mathbf{x} = 0\}.$$

Но $\mathbf{d}^T\mathbf{x} = 0$ тогда и только тогда, когда

$$x_1 = \frac{-\sum_{i=2}^{k} d_i x_i}{d_1}.$$

Для каждого выбора $x_2, \ \ldots, x_k$ правая часть этого равенства фиксирована и равна некоторому элементу v поля **F**. Вероятность того, что x_1 равно v, не превосходит $\frac{1}{2}$, поскольку x_1 равномерно распределено на двухэлементном множестве $\{0,1\}$, а вероятность события $\mathbf{d}^T\mathbf{x} = 0$ либо равна $\frac{1}{2}$, если правая часть принадлежит множеству $\{0,1\}$, либо равна нулю в противном случае. □

Классическим примером задачи, где вероятностные алгоритмы традиционно успешно применяются, является задача проверки тождеств для многочлена от многих переменных.

Задача 17. *Проверить для заданного полинома $P(x_1, \ldots, x_n)$ выполнение тождества $P(x_1, \ldots, x_n) \equiv 0$.*

Следующая лемма (см. [MR95]), по сути, описывает вероятностный алгоритм Монте–Карло с односторонней ошибкой.

Лемма 14. *Пусть $Q(x_1, \ldots, x_n)$ — многочлен от многих переменных степени d над полем F и пусть $S \subseteq F$ — произвольное подмножество. Если $r_1, \ldots, r_n$ выбраны случайно, независимо и равномерно из S, то*

$$\mathsf{P}\left(Q(r_1, \ldots, r_n) = 0 \mid Q(x_1, \ldots, x_n) \not\equiv 0\right) \leq \frac{d}{|S|}.$$

Доказательство. По индукции по n.

Базисный случай $n = 1$ включает полиномы от одной переменной $Q(x_1)$ степени d. Поскольку каждый такой полином имеет не более d корней, вероятность того, что $Q(r_1) = 0$ не превосходит $\frac{d}{|S|}$.

Пусть теперь предположение индукции верно для всех полиномов, зависящих от не более $n - 1$ переменной.

Рассмотрим полином $Q(x_1, \ldots, x_n)$ и разложим его по переменной x_1:

$$Q(x_1, \ldots, x_n) = \sum_{i=0}^{k} x_1^i Q_i(x_2, \ldots, x_n),$$

где $k \leq d$ — наибольшая степень x_1 в Q.

Предполагая, что Q зависит от x_1, имеем $k > 0$, и коэффициент при x_1^k, $Q_k(x_2, \ldots, x_n)$ не равен тождественно нулю. Рассмотрим две возможности.

Первая — $Q_k(r_2, \ldots, r_n) = 0$. Заметим, что степень Q_k не превосходит $d - k$, и по предположению индукции вероятность этого события не превосходит $\frac{(d-k)}{|\mathbf{S}|}$.

Вторая — $Q_k(r_2, \ldots, r_n) \neq 0$. Рассмотрим следующий полином от одной переменной:

$$q(x_1) = Q(x_1, r_2, r_3, \ldots, r_n) = \sum_{i=0}^{k} Q_i(r_2, \ldots, r_n) x_1^i.$$

Полином $q(x_1)$ имеет степень k и не равен тождественно нулю, т.к. коэффициент при x_1^k есть $Q_k(r_2, \ldots, r_n)$. Базовый случай индукции дает, что вероятность события

$$q(r_1) = Q(r_1, r_2, \ldots, r_n) = 0$$

не превосходит $\frac{k}{|\mathbf{S}|}$.

Мы доказали два неравенства:

$$\mathsf{P}\{Q_k(r_2, \ldots, r_n) = 0\} \leq \frac{d-k}{|\mathbf{S}|};$$

$$\mathsf{P}\{Q(r_1, r_2, \ldots, r_n) = 0 \mid Q_k(r_2, \ldots, r_n) \neq 0\} \leq \frac{k}{|\mathbf{S}|}.$$

Используя результат упражнения 4.1.1, мы получаем, что вероятность события $Q(r_1, r_2, \ldots, r_n) = 0$ не превосходит суммы двух вероятностей $(d-k)/|\mathbf{S}|$ и $k/|\mathbf{S}|$, что дает в сумме желаемое $d/|\mathbf{S}|$.

Упражнение 4.1.1. Покажите, что для любых двух событий E_1, E_2,

$$\mathsf{P}\{E_1\} \leq \mathsf{P}\{E_1 | \overline{E}_2\} + \mathsf{P}\{E_2\}.$$

□

Упражнение 4.1.2. Имеется $n \times n$ матрица A, элементами которой являются линейные функции $f_{ij}(x) = a_{ij}x + b_{ij}$.

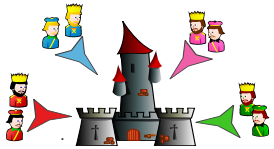
Придумайте алгоритм Монте-Карло с односторонней ошибкой для проверки этой матрицы на вырожденность (det $A \equiv 0$).

Одной из многочисленных областей, где широко применяются вероятностные алгоритмы, являются параллельные и распределенные вычисления. *Эффективным параллельным* алгоритмом (или NC-*алгоритмом*) называется алгоритм, который на многопроцессорной RAM (так называемой PRAM) с числом процессоров, не превосходящим некоторого полинома от длины входа, завершает работу за время, ограниченное полиномом от логарифма длины входа.

Построение эффективного детерминированного параллельного алгоритма (NC-алгоритма) для нахождения максимального паросочетания в двудольном графе является одной из основных открытых проблем в теории параллельных алгоритмов. Удалось, однако, построить эффективный параллельный вероятностный алгоритм нахождения максимального паросочетания в двудольном графе (так называемый RNC-алгоритм) [MR95]. Примеры применения вероятностных алгоритмов для построения эффективных параллельных и распределенных алгоритмов рассматриваются в разделе 4.3.

Доказательство. Вытекает из вышесказанного и результата упражнения 4.3.1. □

4.3.2 Протокол византийского соглашения



Вероятностные методы в построении параллельных и распределенных алгоритмов. Задача о византийских генералах. Распределенный вероятностный протокол византийского соглашения.

В данном разделе мы рассмотрим классическую проблему теории распределенных вычислений о достижении византийского соглашения. Историческая аналогия, давшая название данному протоколу, заключается в выработке соглашения генералами Византии (наступать или не наступать) при условии, что некоторые генералы могут посылать заведомо ложные сообщения (т. е. быть предателями).

Проблема византийского соглашения заключается в следующем (см. [Rab83]). Каждый из n процессоров имеет сначала значение 0 или 1 (в терминологии византийских генералов это соответствует мнению данного генерала «наступать — не наступать»). Пусть b_i — начальное значение i-го процессора. Имеется t ошибочно работающих процессоров, а остальные $n - t$ процессоров рассматриваются как идентичные и исправные. Более того, мы будем предполагать, что неисправные процессоры могут объединять свои усилия по предотвращению достижения соглашения. Предлагаемый протокол должен выстоять против таких атак. Между процессорами возможен обмен сообщениями по правилам, описанным ниже, при которых i-й процессор заканчивает протокол с решением $d_i \in \{0, 1\}$, которое должно удовлетворять следующим требованиям.

Требование 1. *Все исправные процессоры должны завершить протокол с идентичным решением*.

Требование 2. *Если все исправные процессоры начинают протокол обмена с одного и того же значения v, тогда они все должны закончить с общим решением, эквивалентным v.*

Множество неисправных процессоров задается до выполнения протокола, хотя, конечно, исправные процессоры их не знают. Протокол выработки соглашения состоит из нескольких раундов. В течение одного раунда каждый процессор может послать по одному сообщению любому другому процессору. При этом до начала следующего раунда каждый процессор получает сообщения от всех остальных процессоров.

Процессор не обязан посылать одно и то же сообщение остальным процессорам. На самом деле, в описанном ниже протоколе каждое сообщение состоит из одного бита. Все исправные процессоры следуют в точности протоколу. Неисправные процессоры могут посылать сообщения, полностью не соответствующие протоколу, и к тому же могут посылать разные сообщения разным процессорам.

Можно даже считать, что неисправные процессоры перед началом каждого раунда договариваются между собой, какие сообщения послать каким процессорам с целью нанесения наибольшего урона (путаницы).

Соглашение считается достигнутым, если все исправные процессоры вычислили решения, удовлетворяющие двум свойствам, перечисленным выше.

Мы будем изучать число раундов, необходимых для достижения соглашения. Известно, что любой детерминированный протокол достижения соглашения требует не менее $t + 1$ раунда.

Сейчас мы представим простой рандомизированный алгоритм достижения соглашения с математическим ожиданием числа раундов, равным константе. Предполагается, что имеется глобальная «процедура подбрасывания монетки», доступная всем на каждом шаге, которая работает корректно и не подвержена ошибкам. Процедура выдает 1 и 0 независимо с вероятностью каждого исхода $1/2$.

Для простоты будем полагать, что $t < n/8$. Пусть

$$L = 5n/8 + 1, \;\; H = 6n/8 + 1, \;\; G = 7n/8 + 1.$$

На самом деле для правильной работы протокола достаточно, чтобы $L \geq n/2 + t + 1$ и $H \geq L + t$.

В распределенном протоколе i-й ($1 \leq i \leq n$) процессор выполняет алгоритм 32, в котором *coin* обозначает результат подбрасывания монеты (случайный бит).

Подчеркнем, что неисправные процессоры не обязаны следовать протоколу и работают произвольно.

Отметим, что хотя переменная $vote$ каждого процессора и получает некоторое значение на шаге 6, оно не является окончательным и может измениться в дальнейшем. Окончательные значения переменные d_i получают на шаге 7.

Лемма 21. *Если все правильные процессоры начинают раунд с одного и того же значения, то все они принимают решение, равное этому значению, за константное число раундов.*

Более интересен случай для анализа, когда не все правильные процессоры начинают работу с одинакового значения.

В отсутствие ошибочных процессоров для всех процессоров достаточно разослать всем свои значения, после чего все согласовано принимают решение в соответствии с «большинством голосов».

Описанный протокол реализует ту же идею, но в присутствии ошибочных процессоров.

Если два правильных процессора вычисляют разные значения для maj на шаге 3, $tally$ не превосходит $threshold$ вне зависимости от того, было ли выбрано значение для $threshold$ L или H (поскольку разности между $n/2$ и обоими порогами больше t — числа неисправных процессоров).

Тогда все правильные процессоры присваивают $vote$ значение 0 на шаге 6. В результате все правильные процессоры полагают свои решения равными нулю в следующем раунде. Таким образом, остается рассмотреть случай, когда все правильные процессоры вычисляют одно и то же значение для maj на шаге 3.

Скажем, что неисправные процессоры обманывают порог $x \in \{L, H\}$ в некотором раунде, если, посылая различные сообщения правильным процессорам, они вынуждают tally превзойти значение x хотя

Алгоритм 32 Протокол византийского соглашения

Algorithm ByzGen:
Вход: Значение b_i
Выход: Решение d_i

- $vote := b_i$
- Для каждого раунда **выполнить**:
  1. Разослать всем процессорам $vote$.
  2. Получить $votes$ от всех остальных процессоров.
  3. Вычислить maj — самое часто встречающееся значение среди $votes$ (включая собственный).
  4. Вычислить $tally$ — число появлений maj среди полученных $votes$.
  5. **if** $coin = 1$ **then** $threshold := L$
     else $threshold := H$.
  6. **if** $tally \geq threshold$ **then** $vote := maj$
     else $vote := 0$.
  7. **if** $tally \geq G$ **then** присвоить d_i значение maj постоянно.

бы для одного правильного процессора и быть не более x хотя бы для одного правильного процессора. Поскольку разница между двумя порогами L и H больше t, неисправные процессоры могут обмануть не

более одного порога за раунд. Так как порог выбран равновероятно из $\{L, H\}$, обман происходит с вероятностью, не превосходящей $1/2$. Значит, математическое ожидание числа раундов до получения порога без обмана равно 2. Если порог не обманут, то все правильные процессоры вычисляют одно и то же значение v для $vote$ на шаге 6.

В следующем раунде каждый правильный процессор получает по крайней мере $G > H > L$ голосов за v и присваивает maj значение v на шаге 3. Затем на шаге 7 $tally$ превосходит выбранный (любым образом) порог. Когда правильный процессор принимает решение d_i, остальные правильные процессоры должны иметь $tally \geq threshold$, поскольку $G \geq H + t$. Следовательно, они все будут голосовать за одно и то же значение d_i.

Теорема 16. *Математическое ожидание числа раундов для алгоритма **ByzGen** для достижения византийского соглашения ограничено константой.*

4.4 Вероятностное округление

4.4.1 Вероятностное округление для задачи «MAX-SAT»

Рассмотрим следующую задачу ([GW94]).

Задача 20. ***Максимальная выполнимость/MAX-SAT**.*

Даны m скобок конъюнктивной нормальной формы (КНФ) с n переменными. Найти значения переменных, максимизирующие число выполненных скобок.